NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DYNAMIC STALL CALCULATIONS
USING A
NAVIER-STOKES SOLVER

by

James F. Valdes

December 1986

Thesis Advisor Satyanarayana Bodapati
Co-Advisor Lawrence W. Carr

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

SECURITY CLASSIFICATION OF THIS PAGE

AD-A178 566

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable): 31	7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)

Dynamic Stall Calculations Using a Navier-Stokes Solver

12 PERSONAL AUTHOR(S)

VALDES, James F.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1986 December	120

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Dynamic Stall, Navier-Stokes, Unsteady Aerodynamics

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

A Navier-Stokes problem solver, developed by L. N. Sankar, is installed and verified on the ~~NASA Ames~~ Cray X/MP-48 computer and is used to calculate the flow field about a NACA 0012 airfoil oscillating in pitch. Surface pressure distributions and integrated lift, pitching moment, and drag coefficients versus angle of attack are compared to existing experimental data for two cases, involving deep dynamic stall and fully attached flow at and below a freestream Mach number of .3. The flow field about the oscillating airfoil is investigated through the study of contour plots of pressure, density, Mach number, and stream function. The effect of turbulence modeling is explored through use of the Baldwin-Lomax model and a modification designed to prevent underprediction of maximum lift. Finally, Reynolds number and compressibility effects are investigated by repeating the deep stall

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL: Satyanarayana Bodapati	22b TELEPHONE (Include Area Code): 408-646-2854	22c OFFICE SYMBOL: 67Bu

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

19. Abstract (continued)

simulation at one-tenth the experimental Reynolds number and Mach numbers of .3 and .5. The latter conditions are intended for comparison with the results of wind tunnel experiments being planned at NASA Ames Fluid Mechanics Laboratory.

Approved for public release; distribution is unlimited.

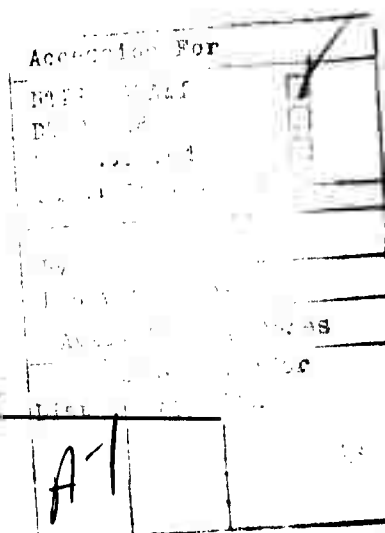
Dynamic Stall Calculations Using a Navier-Stokes Solver

by

James F. Valdes
Lieutenant Commander, United States Navy
B.S., University of Idaho, 1973

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING



from the

NAVAL POSTGRADUATE SCHOOL
December 1986

Author: James F. Valdes
James F. Valdes

Approved by: Satya Bodapati
Satyanarayana Bodapati, Thesis Advisor

Lawrence W. Carr, Co-Advisor

M. F. Platzer, Chairman,
Department of Aeronautical Engineering

John N. Dyer,
Dean of Science and Engineering

ABSTRACT

A Navier-Stokes problem solver, developed by L. N. Sankar, is installed and verified on the NASA Ames Cray X/MP-48 computer and is used to calculate the flow field about a NACA 0012 airfoil oscillating in pitch. Surface pressure distributions and integrated lift, pitching moment, and drag coefficients versus angle of attack are compared to existing experimental data for two cases, involving deep dynamic stall and fully attached flow at and below a freestream Mach number of .3. The flow field about the oscillating airfoil is investigated through the study of contour plots of pressure, density, Mach number, and stream function. The effect of turbulence modeling is explored through use of the Baldwin-Lomax model and a modification designed to prevent underprediction of maximum lift. Finally, Reynolds number and compressibility effects are investigated by repeating the deep stall simulation at one-tenth the experimental Reynolds number and Mach numbers of .3 and .5. The latter conditions are intended for comparison with the results of wind tunnel experiments being planned at NASA Ames Fluid Mechanics Laboratory.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

TABLE OF SYMBOLS

$\mathbf{A}$, $\mathbf{B}$	Jacobian matrices
a	speed of sound
c	chord length
C_d	drag coefficient
C_l	lift coefficient
C_m	pitching moment coefficient
C_p	pressure coefficient
D	Van Driest damping function
e	total energy per unit volume
$\mathbf{F}$, $\mathbf{G}$	flux vectors in the physical plane
$\mathbf{F}^*$, $\mathbf{G}^*$	flux vectors in the transformed plane
I	identity matrix
J	transformation Jacobian
k	reduced frequency
ℓ	eddy viscosity mixing length
M_∞	freestream Mach number
Pr	Prandtl number
p	pressure
$\mathbf{q}$	unknown flow-field vector in the physical plane
$\mathbf{q}^*$	unknown flow-field vector in the transformed plane
Q_x, Q_y	heat-flux vector components
$\mathbf{R}$, $\mathbf{S}$	viscous stress vectors in the physical plane
$\mathbf{R}^*$, $\mathbf{S}^*$	viscous stress vectors in the transformed plane
Re	Reynolds number
U, V	contravariant velocity components
u, v	velocity components in the physical plane
x, y, t	coordinates in the physical plane
z	complex coordinate representation of a point in the physical plane
α	angle of incidence
α_0	the mean angle of incidence
α_1	the amplitude of oscillations

γ	specific heat ratio
$\varepsilon_E, \varepsilon_I$	coefficients of explicit and implicit artificial dissipation terms
μ	viscosity
μ_T	eddy viscosity
ξ, η, τ	coordinates in the transformed plane
$\xi_x, \eta_x, \ldots$	transformation metrics
ρ	density
$\tau_{xx}, \tau_{xy}, \tau_{yy}$	components of shear-stress tensor
ω	vorticity or circular frequency

Operators:

∂	partial derivative
δ	central difference
Δ	incremental change
∇	backward difference or del operator

ACKNOWLEDGEMENTS

The research for this thesis was conducted at the Fluid Mechanics Laboratory of the NASA Ames Research Center under the Navy-NASA Joint Institute of Aeronautics. This work is related to the research program being carried out for the Army Research Office under proposal number 23394-EG. Work was performed under the guidance of Professor Satyanarayana Bodapati, thesis advisor, and Dr. Lawrence W. Carr of NASA Ames, co-advisor. I would like to express my sincere gratitude to them for providing me with the opportunity to become acquainted with the field of computational fluid dynamics and for their patience and encouragement.

I would also like to thank Professor L. N. Sankar of the Georgia Institute of Technology for providing the computer program used and for being so generous with his time during several telephone conversations and, along with his associate Dr. Wei Tang, during a personal visit.

My sincerest appreciation goes to Joan Thompson of Sterling Software for the professional and friendly manner in which she assisted me on occasions too numerous to recall in working with the Ames computer facilities; and to Rosalie Lefkowitz, also of Sterling Software, for her help with graphical presentation, which included providing the carpet plotting routine used for this study.

Finally, I wish to thank my wife, Barbara, for both her patience and her invaluable help in preparation of this thesis, and my daughter, Andrea, for giving up her father during the final few months of this study.

I. INTRODUCTION

Dynamic stall refers to the delay of stall onset beyond static stall angles experienced by airfoils and wings in unsteady motion. The phenomenon first drew serious attention in connection with helicopter aerodynamics when conventional methods of analysis proved incapable of accurately predicting performance for vehicles in high speed forward flight. The observed increase in overall lift could be explained if the lift on the blade moving opposite to flight direction was greater than predicted by steady flow calculations. [Ref. 1] It was experimentally observed that the lifting characteristics of rotors could be adequately modeled by an airfoil oscillating in pitch. The basic mechanism is the shedding of a strong leading-edge vortex which distorts the pressure distribution as it travels over the upper surface of the airfoil and leads to the abrupt changes in lift and pitching moment usually associated with dynamic stall. [Ref. 2]

In addition to helicopter rotors, the dynamic stall phenomenon is observed in such diverse aerodynamic applications as wind turbines, jet engines, and rapidly maneuvering aircraft. Interest in expanding the design envelopes of fighter aircraft through both post stall maneuvers and extended conventional maneuverability has increased interest in dynamic stall research. Progress in this area depends upon improved knowledge of details of viscous flow over airfoils in dynamic stall. [Ref. 1] The basic airfoil dynamic stall process is depicted in Figure 1.1, taken from Reference 1. The flow-field pattern at any instant is critically dependent upon the entire time history, so understanding the chronology of events including time prior to the vortex shedding is crucial. There are four basic stages:

1. trailing-edge flow reversal, progressing forward along the airfoil,
2. formation, growth, and shedding of the vortex,
3. full stall,
4. boundary layer reattachment.

The specific characteristics of a given oscillating airfoil depend primarily on

1. Mach number,
2. Reynolds number,
3. reduced frequency, k, the ratio of the vertical velocity of the leading edge to the freestream velocity, $k = \omega c/2U$, where ω = circular frequency, c = airfoil chord, and U = freestream velocity,

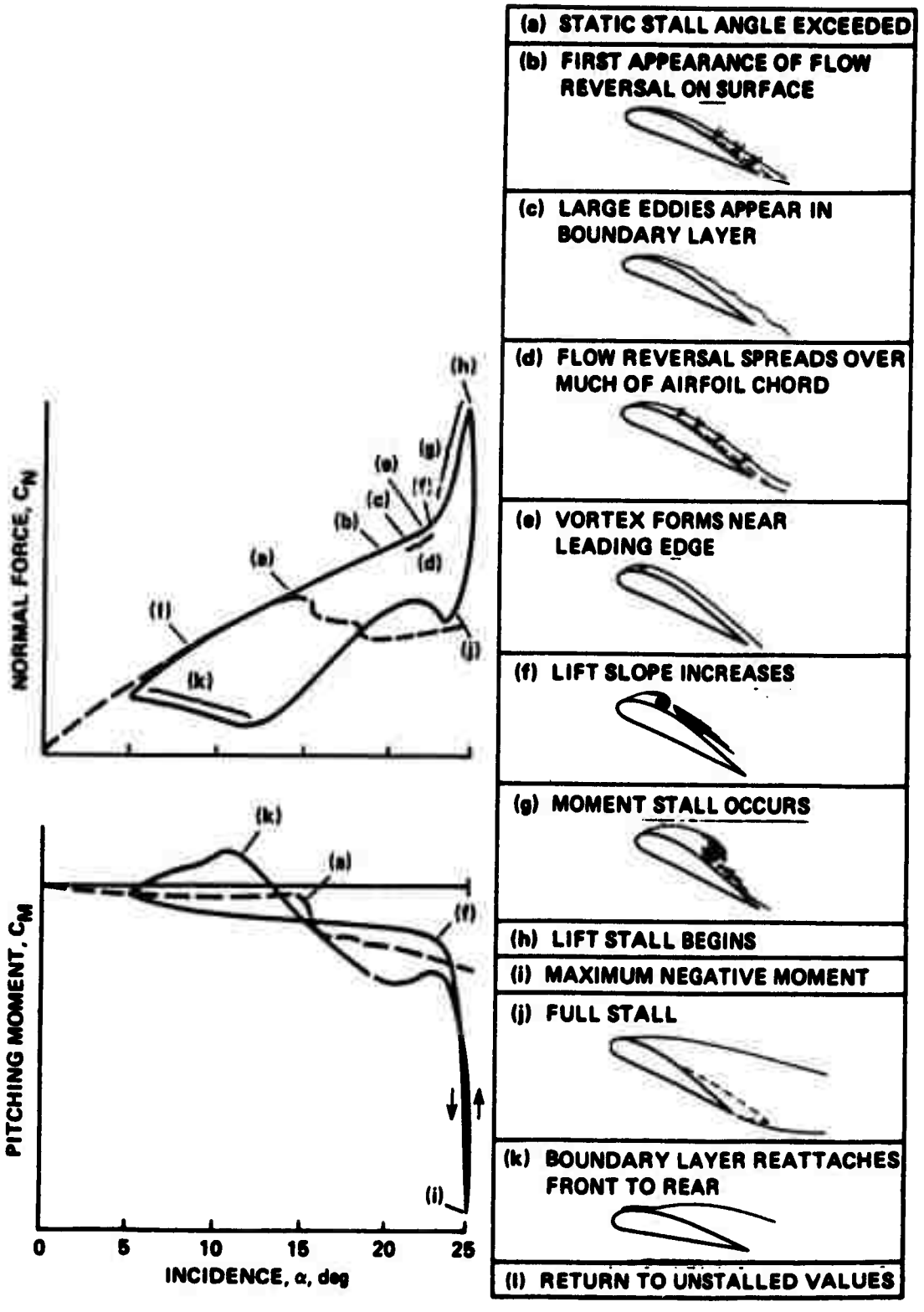


Figure 1.1 Dynamic Stall Chronology (from Carr, Ref. 1).

4. mean angle of attack, α_0,
5. oscillation amplitude, α_1. [Refs. 1,3]

While much of previous research has involved experiments, primarily on airfoils oscillating in pitch, advances in computer technology have increased the potential of computational methods to model details of unsteady flow fields accurately. Early analytic approaches based on inviscid models or semi-empirical analysis were very limited due to the complexity of the dynamic stall phenomenon and its interrelation with viscous effects. Moreover, boundary layer corrections are not appropriate due to the presence of large regions of separation. [Ref. 4] The Navier-Stokes equations can deal with flow separation and shock-boundary layer interaction, and the thin-layer approximation has been successfully applied to airfoils near maximum lift, including--though somewhat less successfully--conditions beyond maximum lift coefficient [Ref. 5]. The thin-layer Navier-Stokes equations may not be appropriate in the case of highly separated flows, however, and in the case of dynamic stall, the viscous layer is of the same order as the airfoil chord during the shedding of the strong leading-edge vortex. The full Navier-Stokes equations must therefore be solved. [Ref. 2]

A Navier-Stokes problem solver has been developed by L. N. Sankar and his associates at the Georgia Institute of Technology and was made available for the present study. An implicit finite-difference procedure is used to solve the two-dimensional, Reynolds-averaged, compressible Navier-Stokes equations in strong conservation form. The flow solver features a moving, body-fitted coordinate system, an algebraic turbulence model (Baldwin-Lomax), and an alternating-direction-implicit (ADI) time-marching algorithm. It can also be used in an inviscid mode to solve the Euler equations. [Refs. 6,7] The goals of the present study were:

1. Develop procedures for exercising the code on the Cray X/MP-48 computer to obtain output in a form suitable for comparing the pressure distribution with experimental results and for investigating details of the computed flow field.
2. Apply the code to steady-state cases and to the conditions of existing dynamic stall experimental data [Ref. 8] and investigate the effect of varying input parameters.
3. Obtain flow-field solutions under proposed conditions of a series of wind tunnel experiments to be conducted at the Fluid Mechanics Laboratory of the National Aeronautics and Space Administration, Ames Research Center.

II. MATHEMATICAL FORMULATION

A. THE SOLUTION DOMAIN

The first step in any numerical solution procedure is to discretize the physical domain. This involves the selection of suitable boundaries and the choice of appropriate nodal points within the boundaries to describe the characteristics of the physical domain accurately. In the solver used here a solution grid is constructed about the airfoil, which is then rotated along with the airfoil for dynamic calculations. With the solution domain defined, it is then necessary to specify conditions at the boundaries. For time dependent problems such as those considered here, initial conditions at the nodal points must also be defined.

1. Algebraic Grid Generation

A mathematical transformation from the physical plane to a computational plane introduces significant advantages for numerical solution. Chief among these is the fact that the physical boundaries are mapped into rectangular surfaces in the transformed plane. Grid points can be concentrated in physical regions experiencing the largest gradients while maintaining the simplicity of uniform spacing in the computational plane. [Refs. 9,10: pp. 519-520] Proper design of the grid is crucial to a stable, accurate solution. The version of Sankar's program used for this study employs an algebraically-generated C-grid. This grid is quickly produced, easily rotated by simple sine/cosine relationships with the angle of attack, and allows a high degree of clustering in the normal direction to cover adequately the thin boundary layer of high Reynolds number flows.

The procedure used generates a sheared parabolic coordinate system based on an input airfoil shape. Two points are first found on the airfoil: point N halfway between the nose and the center of curvature of the nose, and point T at the trailing edge. The cut along the airfoil wake is chosen to be tangent to the mean camber line at the trailing edge. The airfoil shape and wake are then unwrapped to form a surface S(X) in an intermediate X-Y plane using the following transformation:

$$X + iS(X) = \sqrt{z - z_N} \qquad \text{(eqn 2.1)}$$

where $z = x + iy$ locates a point in the physical (x-y) plane. This surface will then lie along the X-axis with a shallow bump at the origin which coincides with the singular point z_N. Figure 2.1 illustrates this mapping for a symmetric airfoil. Cubic interpolation is used to identify additional points and smooth the surface S(X). Lines parallel to the Y-axis and lines equidistant from the surface S(X) are then constructed before the sheared cartesian grid is mapped back to the physical plane. Each point in the physical domain is assigned to a corresponding point in the computational (ξ-η) plane through the intermediate plane relationship. The intermediate and computational planes are illustrated in Figure 2.2. Unit spacing is assumed in the computational plane, simplifying calculations. For solutions of viscous flows, the ξ = constant lines are retained while the η = constant lines in the physical plane are redistributed to place points within the boundary layer. The first point off the solid surface is placed at a distance specified by the user and the remaining lines are exponentially distributed to the far-field boundary. [Refs. 6,7: pp. 40-44]

The final mesh used is shown in Figure 2.3. The number of points defined in the ξ-direction is 161 of which 159 are used for calculations, and 41 points are defined in the η-direction. A detail of the grid around the airfoil is shown in Figure 2.4. The spacing along constant-ξ lines is relatively fine near the leading edge, where the sharpest gradients are expected, and coarse at the trailing edge. In order to perform calculations in the computational plane, a transformation is required to map each point to the corresponding point in the physical plane. In the present solver, a numerical approach is taken: finite differences are used to relate the mesh spacings in the two planes through transformation metrics. This method is outlined in later sections.

2. **Initial Conditions and Boundary Conditions**

The initial conditions for steady-state calculations are assumed to be the freestream conditions, and inaccuracies introduced by this crude starting approximation are removed by viscous effects after the airfoil is impulsively started. (Inviscid calculations require proper boundary conditions and artificial dissipation to correct the solution). [Ref. 7: p. 13] In the case of dynamic calculations, the initial conditions are obtained from an asymptotically converged steady-state solution at the minimum airfoil angle of incidence. The solution is then marched in time as the oscillation cycle begins from this angle. Boundary conditions are explicitly applied at each time step on the solid boundary, at the far-field boundary, and in the airfoil wake, where the grid

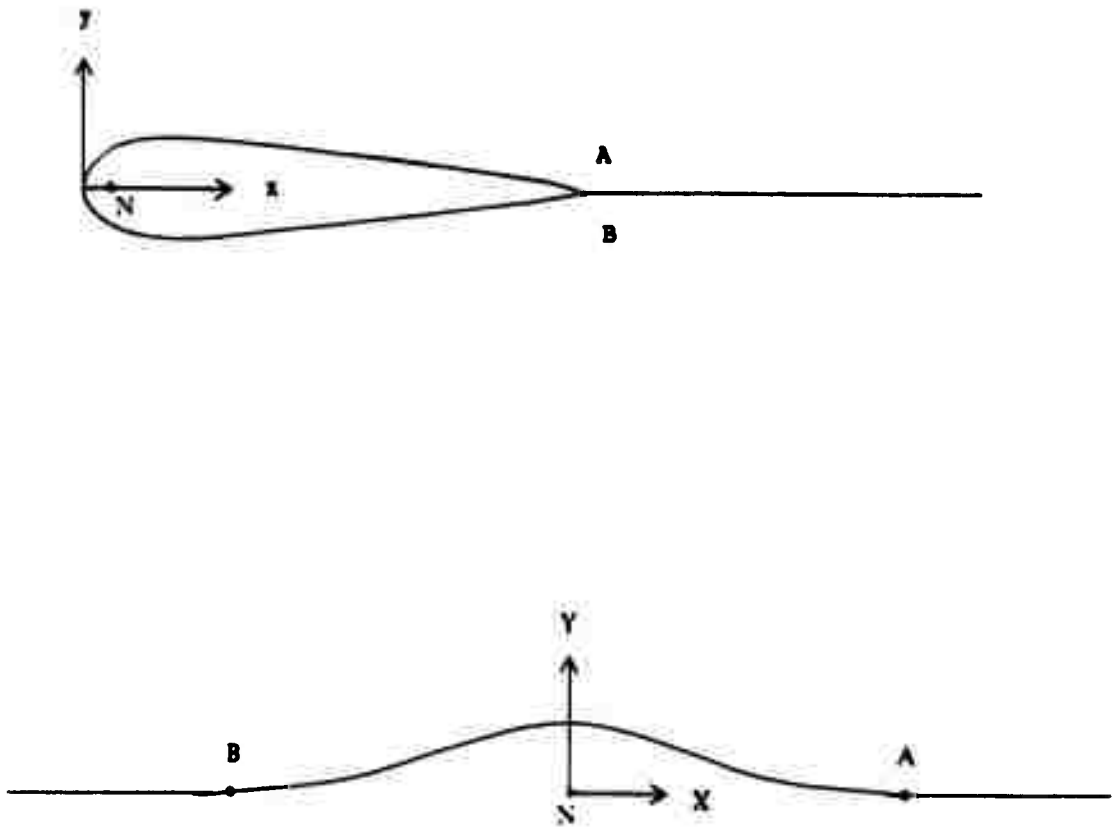


Figure 2.1 Symmetric Airfoil in the Physical Plane (top)
Unwrapped in the Intermediate Plane (bottom).

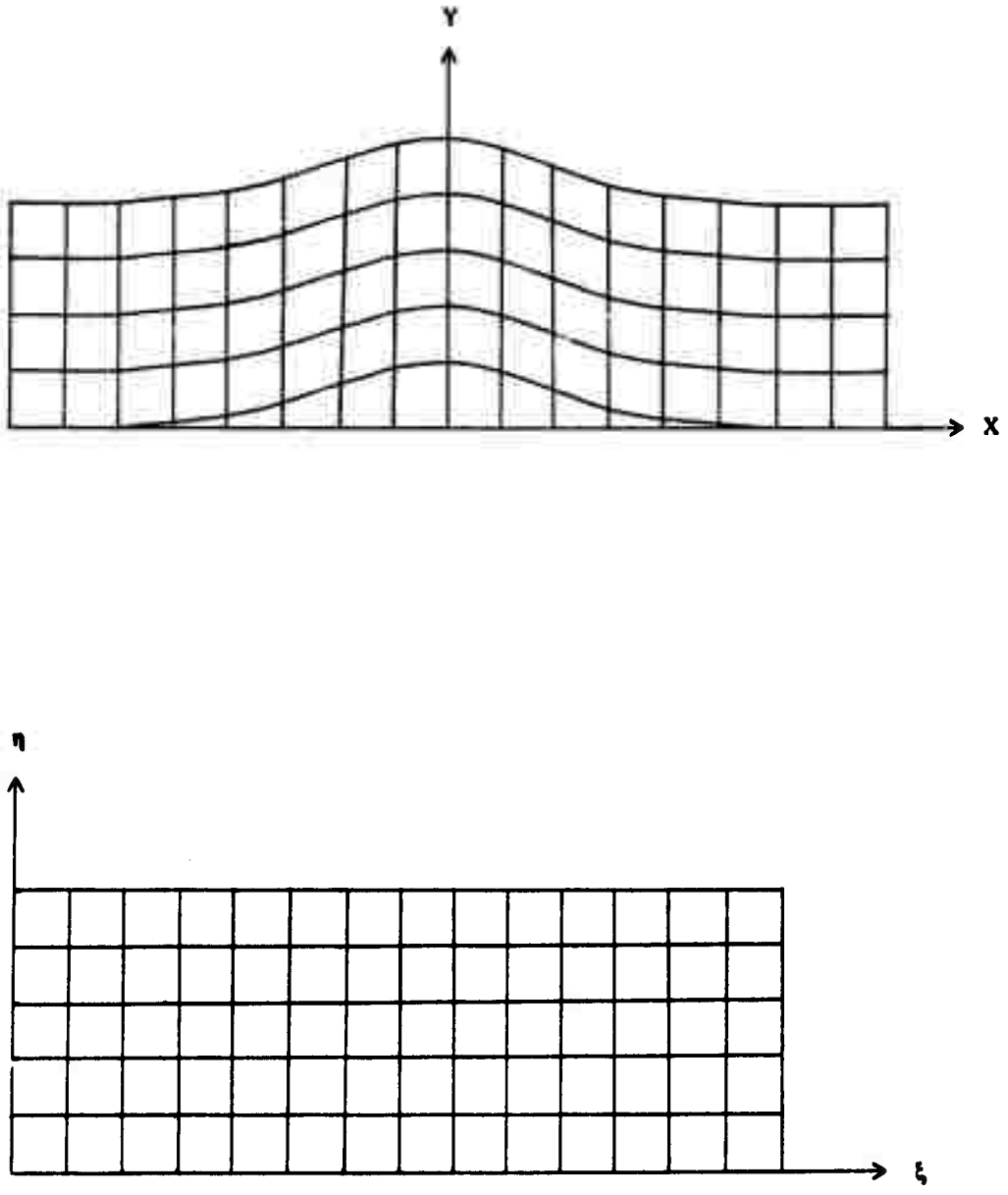


Figure 2.2 Construction of the Grid in the Intermediate Plane (top) Corresponding Computational Plane Grid (bottom).

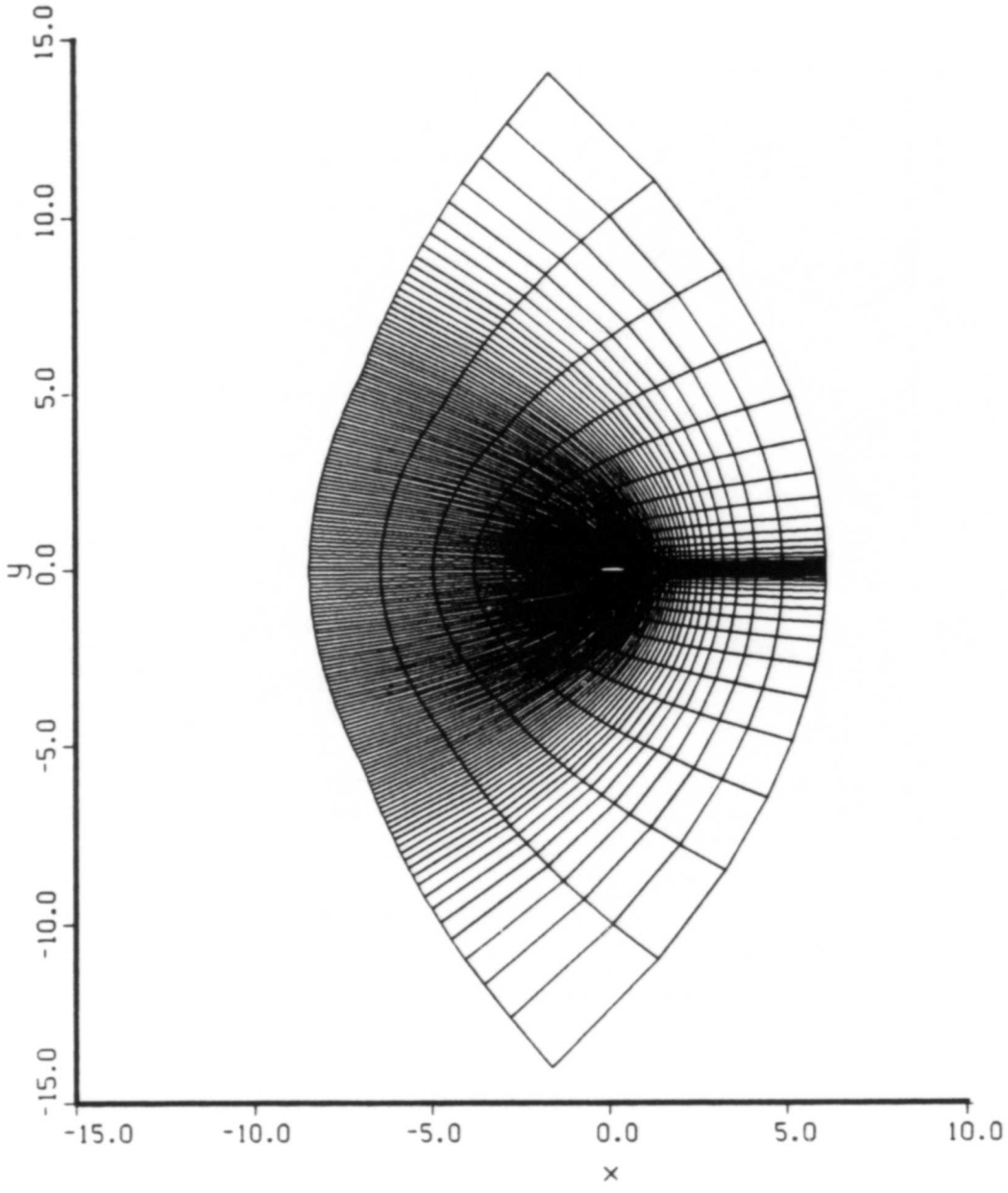


Figure 2.3 Algebraic C-Grid in the Physical Plane.

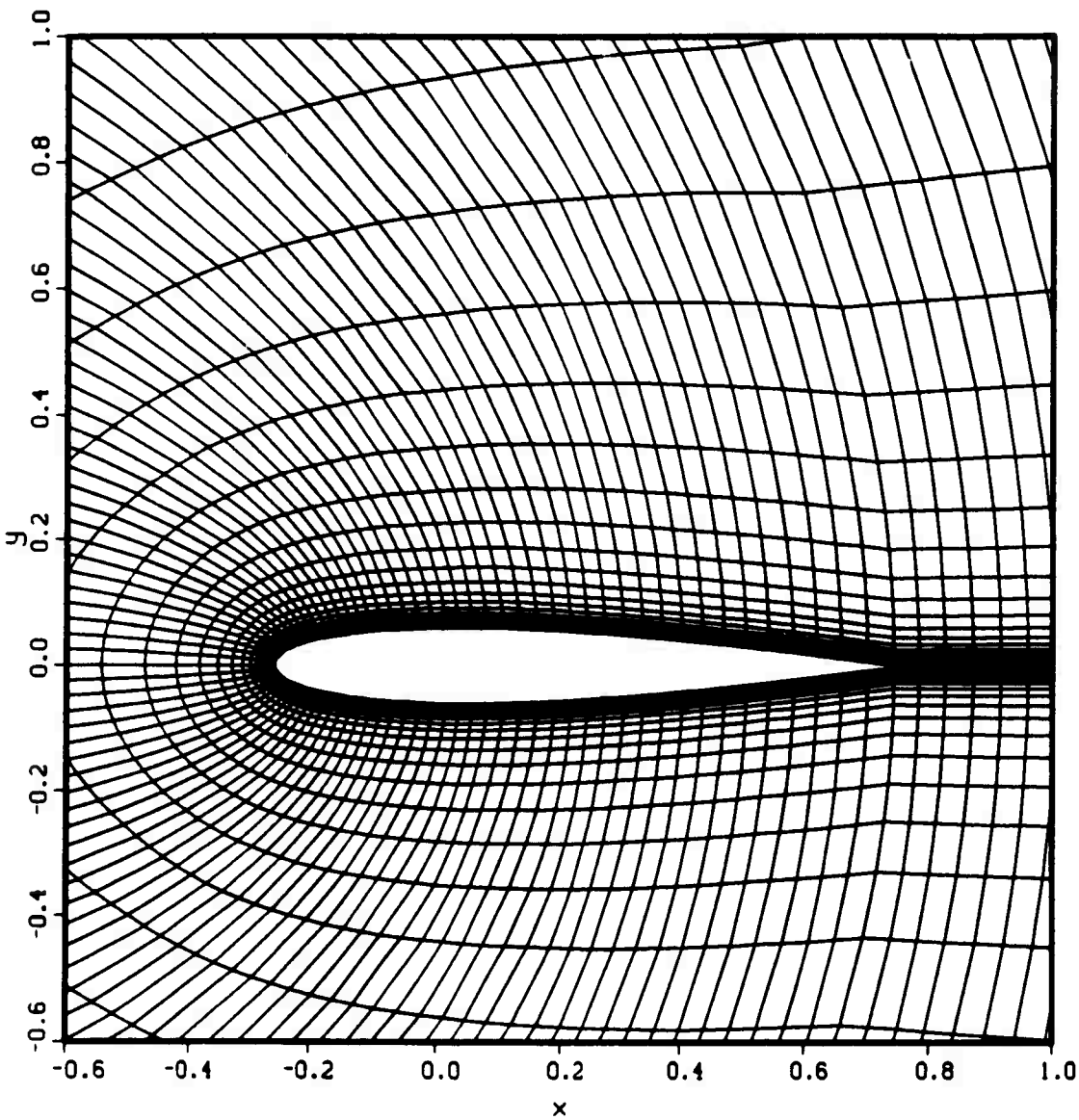


Figure 2.4 Detail of Grid near the Airfoil.

generation procedure introduced a "cut" in the physical domain between the airfoil trailing edge and the downstream boundary. (See Fig. 2.3) [Refs. 6,11]

B. GOVERNING EQUATIONS

The unsteady, compressible, two-dimensional Navier-Stokes equations are written in cartesian coordinates as:

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{P} = 0,$$

$$\mathbf{P} = \begin{bmatrix} \rho u \\ \rho u^2 + p - \tau_{xx} \\ \rho uv - \tau_{xy} \\ (e+p)u - u\tau_{xx} - v\tau_{xy} + Q_x \end{bmatrix} \mathbf{i} + \begin{bmatrix} \rho v \\ \rho uv - \tau_{xy} \\ \rho + p - \tau_{yy} \\ (e+p)v - v\tau_{yy} - u\tau_{xy} + Q_y \end{bmatrix} \mathbf{j},$$

$$\mathbf{q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix} \qquad \text{(eqn 2.2)}$$

where ρ, u, v, and e are density, velocity components, and total energy per unit volume; τ_{xx}, τ_{yy}, and τ_{xy} are components of the shear-stress tensor derived by application of Stokes' hypothesis; and Q_x and Q_y are the heat-flux vector components given by:

$$Q_i = -k\partial_i T = -\frac{\mu}{(\gamma-1)Pr}\partial_i(a^2) \qquad \text{(eqn 2.3)}$$

in which μ is the coefficient of viscosity, Pr is the Prandtl number, and a is the speed of sound. [Refs. 12,10: pp. 480-481] The specific heat ratio, γ, is assumed to be 1.4. Expanding and rearranging Equation 2.2 yields [Refs. 6,7: pp. 9-10]:

$$\partial_t \mathbf{q} + \partial_x \mathbf{F} + \partial_y \mathbf{G} = \partial_x \mathbf{R} + \partial_y \mathbf{S} \qquad \text{(eqn 2.4)}$$

with

$$\mathbf{q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix} \quad \mathbf{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (e+p)u \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho + p \\ (e+p)v \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ R_4 \end{bmatrix} \quad \mathbf{S} = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ S_4 \end{bmatrix}$$

and

$$R_4 = u\tau_{xx} + v\tau_{xy} + K\partial_x(a^2)$$
$$S_4 = u\tau_{xy} + v\tau_{yy} + K\partial_y(a^2)$$
$$K = \frac{\mu}{(\gamma-1)Pr} \qquad \text{(eqn 2.5)}$$

In this form **F** and **R** contain the flux and viscous stress terms in the x-direction and **G** and **S** the flux and viscous stress terms in the y-direction. Note that the viscous terms are contained on the right-hand side (setting them equal to zero produces the Euler equations). The governing equations are given in strong conservation form. That is, the coefficients of the derivative terms are constant, and the equations may be written in a form expressing the divergence of the physical quantities of mass, momentum, and energy (Eqn. 2.2). Such equations may be expressed in a corresponding integral form through the divergence theorem:

$$\int_V \nabla \cdot \mathbf{P} dv = 0 = \int_S \mathbf{P} \cdot \mathbf{n} ds \qquad \text{(eqn 2.6)}$$

The differential statement of the governing equations is not valid across discontinuities (ie., shocks); however the integral expression is. [Ref. 13: pp. 406-408] Using the conservative form means that the finite-difference approximation will ensure conservation of the physical properties and thus satisfy the corresponding integral form of the governing equations. [Refs. 14,10: pp. 50-52] This gives *weak solutions* (solutions of partial differential equations that include discontinuities) [Ref. 10: pp. 139-141] in the vicinity of shocks. Such methods are called *shock capturing*, and their use is important in treatment of the compressible dynamic stall problem.

As discussed earlier, considerable simplification of the numerical solution may be realized by employing a transformation into a rectangular plane. It can be shown that the strong conservation-law form may be restored following such a transformation [Refs. 14,9: p. 254]. The general transformation is given by: [Ref. 6]

$$\begin{aligned} \xi &= \xi(x,y,t) \\ \eta &= \eta(x,y,t) \\ \tau &= t \end{aligned} \qquad \text{(eqn 2.7)}$$

and the *Jacobian* and *metrics* of the transformation by:

$$J = \xi_x \eta_y - \eta_x \xi_y = 1/(x_\xi y_\eta - x_\eta y_\xi) \qquad \text{(eqn 2.8)}$$

$$\begin{aligned} \xi_x &= J y_\eta \\ \xi_y &= -J x_\eta \\ \xi_t &= -x_\tau \xi_x - y_\tau \xi_y \\ \eta_x &= -J y_\xi \\ \eta_y &= J x_\xi \\ \eta_t &= -x_\tau \eta_x - y_\tau \eta_y \end{aligned} \qquad \text{(eqn 2.9)}$$

Here the subscripts indicate partial differentiation with respect to the subscripted variable. The Jacobian represents the magnification factor of mesh elements between the physical and computational planes [Ref. 10: p. 530]. The time- and spatial-derivatives of any quantity q are given by:

$$\begin{aligned} q_t &= q_\tau + q_\xi \xi_t + q_\eta \eta_t \\ q_x &= q_\xi \xi_x + q_\eta \eta_x \end{aligned} \qquad \text{(eqn 2.10)}$$

The governing equation (Eqn. 2.4) in the transformed plane may then be expressed as:

$$\partial_\tau q^* + \partial_\xi F^* + \partial_\eta G^* = \partial_\xi R^* + \partial_\eta S^* \qquad \text{(eqn 2.11)}$$

where

$$q^* = q/J$$

$$F^* = (\xi_x F + \xi_y G + \xi_t q)/J$$

$$G^* = (\eta_x F + \eta_y G + \eta_t q)/J$$

$$R^* = (\xi_x R + \xi_y S)/J$$

$$S^* = (\eta_x R + \eta_y S)/J \qquad \text{(eqn 2.12)}$$

This constitutes a coupled, highly non-linear system of equations. The flux vectors **F*** and **G*** may be stated in a less complex form by introducing the contravariant velocity, which is required to compensate for grid motion. The contravariant velocity components are given by [Ref. 9]:

$$U = \xi_t + \xi_x u + \xi_y v = \xi_x(u - x_\tau) + \xi_y(v - y_\tau)$$

$$V = \eta_t + \eta_x u + \eta_y v = \eta_x(u - x_\tau) + \eta_y(v - y_\tau) \qquad \text{(eqn 2.13)}$$

Using these velocities, **F*** and **G*** may be rewritten, as used in the present implementation:

$$F^* = \frac{1}{J}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ (e+p)U - \eta_t p \end{bmatrix} \quad G^* = \frac{1}{J}\begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ (e+p)V - \eta_t p \end{bmatrix} \qquad \text{(eqn 2.14)}$$

C. NUMERICAL SOLUTION

The finite-difference scheme employed in Sankar's problem solver is based on the Beam-Warming approximate factorization algorithm [Ref. 10: pp. 489-496] as implemented by Steger [Ref. 9]. A theoretical description is given in References 6 and 7. Solving the governing equation in the transformed plane (Eqn. 2.11) requires determination of the metrics and Jacobian, relating the variables to corresponding quantities in the physical plane through Equations 2.12 and 2.14. This is accomplished through Equations 2.8 and 2.9, where central difference approximations are used to

compute the spatial derivatives x_ξ, y_ξ, x_η, and y_η at each interior grid point, and x_τ and y_τ are the grid velocity components. One-sided differences are used at the boundaries. The governing equation (Eqn. 2.11) is parabolic (hyperbolic when the viscous terms are neglected) [Ref. 10: p. 139], lending itself naturally to an implicit solution approach [Ref. 14]. This allows the use of a larger time step than explicit schemes, and the requirements on step size are generally set by accuracy rather than stability requirements [Refs. 2,9]. The use of explicit boundary conditions in the present implementation introduces a step-size stability requirement. Given an assumed flow field at time t_n, an alternating-direction-implicit (ADI) procedure is used to advance the solution to time level t_{n+1}. Elements of the vector $\mathbf{q}^*$ are then given in terms of values at time t_n and increments, ie. $\{\mathbf{q}^*\}^{n+1} = \{\mathbf{q}^*\}^n + \{\Delta\mathbf{q}^*\}^{n+1}$. In Sankar's program, the approximation method used is an Euler implicit finite-difference scheme [Ref. 10: p. 98], with central spatial differences and backward time differences for the derivatives in the governing equation. The method is thus first-order time accurate and second-order accurate in space. [Ref. 7: pp. 21-23]

1. **Linearization and Factorization**

Inspection of the terms of the flux and viscous stress vectors of the governing equation shows them to be very non-linear functions of the unknown vector $\mathbf{q}^*$. The solution procedure involves the use of truncated Taylor series expansions to linearize the equation [Ref. 10: pp. 490-491]. The viscous terms, however, are treated explicitly, evaluated from the solution at the previous time level, rather than following Steger's fully implicit treatment [Ref. 9]. This leads to improved computational efficiency but requires implicit smoothing for stability at moderate to high Reynolds numbers [Ref. 6]. The Euler implicit form may be obtained from Taylor series expansion at time t_{n+1}. Replacing the partial time derivative with the backward difference operator ∇_t yields:

$$\{\Delta\mathbf{q}^*\}^{n+1} = \{\mathbf{q}^*\}^{n+1} - \{\mathbf{q}^*\}^n = \Delta t(\nabla_t\{\mathbf{q}^*\})^{n+1} + O(\Delta t)^2 \qquad \text{(eqn 2.15)}$$

Substituting Equation 2.11 into Equation 2.15, after replacing the spatial derivatives

with the central difference operators δ_ξ and δ_η and the time derivative with the backward difference operator, produces the governing equation in non-linear, time-differenced form [Refs. 7,10: pp. 22-23,490-491]:

$$\{\Delta q^*\}^{n+1} = -\Delta t(\delta_\xi\{F^*\}^{n+1} + \delta_\eta\{G^*\}^{n+1}) + \Delta t(\delta_\xi\{R^*\}^{n+1} + \delta_\eta\{S^*\}^{n+1}) + O(\Delta t)^2 \qquad \text{(eqn 2.16)}$$

Taylor series expansions of the flux vectors are now introduced:

$$\{F^*\}^{n+1} = \{F^*\}^n + [\partial_q F^*]^n\{\Delta q^*\}^{n+1} + O(\Delta t)^2$$
$$\{G^*\}^{n+1} = \{G^*\}^n + [\partial_q G^*]^n\{\Delta q^*\}^{n+1} + O(\Delta t)^2 \qquad \text{(eqn 2.17)}$$

Here $[\partial_q F^*]^n$ and $[\partial_q G^*]^n$ are Jacobian matrices, evaluated under the assumption of a perfect gas [Refs. 7,10: pp. 87-88,491], which will be denoted [A] and [B] respectively. Substituting Equation 2.17 into 2.16 and placing the implicit terms on the left side and explicit terms on the right side gives the linearized system

$$([I] + \Delta t\delta_\xi[A] + \Delta t\delta_\eta[B])\{\Delta q^*\}^{n+1} = \{R\}^n \qquad \text{(eqn 2.18)}$$

where

$$\{R\}^n = -\Delta t(\delta_\xi\{F^*\}^n + \delta_\eta\{G^*\}^n) + \Delta t(\delta_\xi\{R^*\}^{n+1} + \delta_\eta\{S^*\}^{n+1})$$

Since, as mentioned previously, the viscous terms are treated explicitly, they have been grouped together in $\{R\}^n$ with the other terms to be evaluated at the known time level t_n. [Refs. 6,7: pp. 23-24]

Although linearized, the system of equations is now in block pentadiagonal form and still computationally expensive to solve. *Approximate factorization* is used to write the implicit scheme as a product of one-dimensional factors so that the problem may be solved by a sequence of relatively simple operations, subject to the additional error introduced by the factorization approximation. [Ref. 14] Factoring the left side of Equation 2.18 then gives [Ref. 7: p. 25]

$$([I] + \Delta t\delta_\xi[A])([I] + \Delta t\delta_\eta[B])\{\Delta q^*\}^{n+1} = \{R\}^n \qquad \text{(eqn 2.19)}$$

The Beam-Warming algorithm is now implemented in three steps [Ref. 10: p. 494]:

Step 1

$$([I] + \Delta t \delta_\xi [A])\{\Delta q^{**}\} = \{R\}^n \qquad \text{(eqn 2.20)}$$

Step 2

$$([I] + \Delta t \delta_\eta [B])\{\Delta q^{*}\}^{n+1} = \{\Delta q^{**}\} \qquad \text{(eqn 2.21)}$$

Step 3

$$\{q^{*}\}^{n+1} = \{q^{*}\}^{n} + \{\Delta q^{*}\}^{n+1} \qquad \text{(eqn 2.22)}$$

Notice that steps one and two represent sweeps in alternating ξ- and η- directions, leading to the name alternating-direction-implicit for this method. Since the difference operators δ_ξ and δ_η represent central difference approximations, they lead to systems of block tridiagonal matrix equations composed of 4×4 submatrices, which are easily solved.

2. **Application of Boundary Conditions**

On the solid surface the flow tangency condition applies for the Euler equations, and in addition, the no-slip condition applies for the viscous flows considered here. This means the fluid and solid surface must have the same velocity at the common boundary, which is to say the *contravariant velocity* components U and V (Eqn. 2.13) are both zero. The surface is assumed adiabatic so $\partial_n e = 0$. It is also assumed, as reasonable approximations for high Reynolds number flows, that $\partial_n p = 0$ and $\partial_n \rho = 0$ at the surface. Pressure and density at the surface are then numerically determined by two-point extrapolation, $p_{i,1} = (4p_{i,2} - p_{i,3})/3$ and $\rho_{i,1} = (4\rho_{i,2} - \rho_{i,3})/3$ internal energy is calculated using the relation $p = (\gamma - 1)(e - 0.5\rho(u^2 + v^2))$. Boundary conditions are specified following each ADI sweep for the incremental quantities $\{\Delta q^{*}\}$ and $\{\Delta q^{**}\}$ by setting them equal to zero on the solid surface. [Refs. 6,7,11] The far-field boundary conditions are undisturbed freestream conditions. Conditions along the cut in the C-grid are found by averaging the solution at the nearest interior points on either side of the cut. [Ref. 7: pp. 27-29] The effect of this scheme is to update conditions at the boundaries explicitly. The explicit boundary conditions are used for their simplicity in the code, although implicit conditions would generally be preferable for stability and accuracy of the numerical solution [Refs. 9,15].

3. Artificial Dissipation

When Taylor series expansions are substituted into the finite-difference approximation of a partial differential equation, and the terms are rearranged to produce the original differential equation plus a truncation error, the lowest order truncation error term may contain a second derivative that makes the term similar in appearance to the viscous terms in flow equations. Thus the use of finite-difference approximations introduces an "artificial viscosity" into the solution. [Ref. 10: pp. 89-92] One-sided difference schemes (upwind differencing) are commonly used to produce this *implicit dissipation*. Additionally, terms may be added explicitly to suppress oscillatory behavior of the numerical solution. Central difference approximations, given by $(u_{i+1} - u_{i-1})/2\Delta x$, tend to decouple the odd from the even terms, since only every other term is used in the difference formula. For example, the sequence of nodal values $\{-1,+1,-1,+1,...\}$ would give a central difference approximation to the first derivative of zero. [Ref. 14] Solutions of the Navier-Stokes equations may "blow up" due to numerical oscillations if the mesh is not fine enough in areas of large pressure gradients. To suppress high frequency numerical oscillations, fourth-order *explicit dissipation* terms are commonly added to algorithms. [Ref. 10: pp. 105, 486] Physical dissipation diffuses energy, and artificial dissipation reduces gradients in the flow-field solution whether such diffusion is physically correct or numerically induced. The explicit method of adding dissipation has the advantage over one-sided differencing of making the physical approximations clearer and permitting some control over the amount of non-physical (numerical) dissipation. [Ref. 4] Although viscous terms provide a dissipative mechanism, some additional numerical dissipation is always necessary; the degree should be kept to the minimum required. [Refs. 14,10: p. 92]

The Navier-Stokes code compiled by Sankar follows Steger in the use of artificial dissipation, with changes to prevent overshooting in the vicinity of shocks. [Ref. 16] Dissipation is employed for four main purposes, the first three of which have been mentioned previously:

1. Suppress high frequency oscillations in the solution caused by the use of central differences.
2. Correct for the incorrect initial conditions, especially when solving inviscid flows.
3. Allow explicit treatment of viscous terms.
4. Alleviate restrictive stability bounds on the use of explicit damping. [Ref. 9]

The first two reasons pertain to the right-hand (explicit) side of Equations 2.19 and 2.20 and the second two reasons to the left-hand (implicit) side of Equations 2.19, 2.20, and 2.21. The explicit dissipation used is a combination of second- and fourth-order terms. Fourth-order smoothing is normally used for accuracy, since second-order terms tend to smear rapid pressure variations near the leading edge. In large pressure gradients such as shocks, however, these terms lead to overshooting in the pressure distribution. To prevent this, the second derivative of pressure is used to control the degree of dissipation, turning on the second-order term and suppressing the fourth-order term in the vicinity of shocks [Ref. 6].

Upwind difference schemes may be rewritten as the sum of central difference approximations to the first and second derivatives as follows:

$$(u_i - u_{i-1})/\Delta x = (u_{i+1} - u_{i-1})/2\Delta x - (u_{i+1} + 2u_i - u_{i-1})/2\Delta x \qquad \text{(eqn 2.23)}$$

The second term on the right-hand side may then be regarded as an implicit dissipation term, giving a central difference scheme the artificial viscosity characteristics of upwind differencing. Second-order implicit smoothing terms are added to the left-hand side of Equations 2.19, 2.20, and 2.21, both to allow the use of explicitly evaluated viscous terms and to permit the use of larger magnitude explicit damping terms without loss of stability. The dissipation terms are of at most the same order as the truncation errors of the finite-difference approximation involved [Refs. 6,7: p. 30-33]. The coefficients of both implicit and explicit dissipation terms, ε_I and ε_E, are user selectable options that control the magnitude of the artificial viscosity.

D. TURBULENCE MODEL

The Reynolds form of the Navier-Stokes equations introduces new unknowns in the form of turbulent stress and heat-flux terms, requiring additional relations to make solution of the system possible. This is known as the *closure problem*, and the approach usually taken to resolve it is *turbulence modeling*. [Ref. 10: pp. 207-208, 221-235] Under the Boussinesq assumption, the coefficients of viscosity and thermal conductivity are replaced by the combinations $\mu + \mu_t$ and $k + k_t$ representing the sum of laminar and turbulent components. In practice the thermal conductivity is usually determined from the relationships $k = c_p\mu/Pr$ and $k_t = c_p\mu_t/Pr_t$. In the solver used here a constant value of $Pr = 1$ is assumed, and total viscosity is used in Equation 2.3

rather than directly calculating the conductivity. For the present calculations, the viscous work and conductivity terms R_4 and S_4 (Eqn. 2.5) were set equal to zero. The laminar viscosity is easily found from the Reynolds number and freestream velocity, and an algebraic turbulence model is used to determine the turbulent eddy viscosity.

Turbulence modeling lies at the frontier of research in computational fluid dynamics. The use of Navier-Stokes solvers as practical, predictive tools depends on the development of adequate turbulence models. For many uses, however, the dissipative, diffusive nature of turbulence has enabled simple algebraic models to perform beyond their theoretical limitations [Ref. 17]. In the present case the complex, history-dependent viscous effects dominating the flow are clearly beyond the capacity of simple algebraic models based strictly on local flow properties. Unfortunately, there are no completely satisfactory alternatives. The Baldwin-Lomax model was chosen for simplicity, while acknowledging that it might be unsuitable for the massively separated flows experienced in dynamic stall [Ref. 6]. When it was discovered that the maximum lift was underpredicted, a modification was incorporated which will be discussed following a description of the original Baldwin-Lomax model.

1. **Baldwin-Lomax Model**

The Baldwin-Lomax model is based on Cebeci's two-layer model, with μ_t given by the inner-layer formulation out to the smallest normal distance at which the inner- and outer-layer formulas give the same value for the viscosity. In the inner layer a simple mixing length formula is used with the eddy viscosity proportional to the magnitude of the local vorticity times the square of the normal distance from the surface:

$$(\mu_T)_{inner} = \rho \ell^2 |\omega|$$

where the mixing length

$$\ell = \kappa y D \qquad \text{(eqn 2.24)}$$

and D is the Van Driest damping function, given by:

$$D = [1 - \exp(-y^+ / A^+)] \qquad \text{(eqn 2.25)}$$

The damping constant $A^+ = 26$, and the von Karman constant, κ, is taken as 0.4. In the outer layer, the locally constant eddy viscosity is given by:

$$(\mu_T)_{outer} = KC_{cp}\rho F_{wake} F_{Kleb}(y) \qquad \text{(eqn 2.26)}$$

The key feature is the definition of a function

$$F_{wake} = \min(y_{max}F_{max}, C_{wk}y_{max}U_{dif}^2/F_{max}) \qquad \text{(eqn 2.27)}$$

where F_{max} is the maximum of the function

$$F(y) = y|\omega|D \qquad \text{(eqn 2.28)}$$

and y_{max} is the value of y at which F_{max} occurs. The Klebanoff intermitancy function F_{Kleb}, has the effect of causing the eddy viscosity to approach zero in the far field. U_{dif} is the difference between the maximum and minimum velocities in the velocity profile. The values of the constants used are $K = 0.0168$, $C_{cp} = 1.6$, and $C_{wk} = 0.3$. [Refs. 2,6,7: pp. 16-19]

2. **Modified Model**

It has been observed that algebraic turbulence models of the Cebeci-Smith type [Ref. 10: p. 225] perform well in attached flows at moderate angles of attack but overpredict both the rise in turbulent shear stress in adverse pressure gradients, and the stress decrease when pressure gradients are relieved [Ref. 18]. The Baldwin-Lomax model was found by Sankar to cause an early prediction of flow separation at high angles of attack. It was assumed that this was due to underprediction of the turbulent length scale at high angles. The function F, defined in Equation 2.28, has in some cases been found to possess a number of local maxima, leading to large length scale variations depending on the maximum chosen. This presented a possibility for simple modification to increase the outer-layer velocity and length scales. The definition of F_{max} was redefined to be the value of the function F when the *product* yF(y) is maximum. In preliminary investigation this change was found to increase stall angles but to lead to premature reattachment during the downstroke in dynamic calculations. For this reason its use is recommended only until stall onset. [Ref. 19]

III. DESCRIPTION OF THE CODE

The version of Sankar's code used in the present study is presented in Appendix A. Appendix B contains instructions and guidelines for running the program. Modifications were made in the output scheme to produce output of pressure coefficients and flow-field plotting data at a specified number of equal time intervals during each cycle, and job cards were prepared for save, restart, and various output options. Other features of the program were not altered, although comments were added. Subroutines in the program supplied by Sankar were fully vectorized for the Cray computer with the exception of subroutine EDDY. Execution times on the NASA Ames Research Center's Cray X-MP/48 computer, for full viscous dynamic calculations, are approximately 0.318 seconds per time step.

All program variables are nondimensionalized. This has the effect of multiplying the right side of the governing Equation 2.11 by a factor of Re^{-1} but does not affect the basic form of the equation [Ref. 10: pp. 191-193]. The nondimensionalization is effected as follows:

1. density with respect to ρ_∞
2. length with respect to c
3. time with respect to c/a_∞
4. velocities with respect to a_∞
5. total energy with respect to $\rho_\infty a_\infty^2$

Under this scheme the number of time steps required to complete an oscillation cycle is given by $\pi/(k \times M_\infty \times \Delta t)$.

A. MAIN PROGRAM

The program begins execution by reading the input data. This consists of the dynamic stall parameters, α_0, α_1, k, M, and Re; program control information, including time-step size, explicit viscosity coefficient, and flags for features such as the modified Baldwin-Lomax turbulence model; and the airfoil coordinates and related information. Table 1 summarizes the subroutine calls. The grid generation routine, AIRFOL, is called, and for viscous flows, CLUSTR is then called to recompute the constant-η grid lines so that the first point from the airfoil surface is at the user-specified distance. The initial conditions are established next, and if the program is being restarted from

previous calculations, the values of time and the unknown flow-field vector q^* are replaced with those now read from logical unit 7. Finally, subroutine METRIC is called to compute the metrics and transformation Jacobian, and the iterative calculations begin.

TABLE 1
SUBROUTINE CALLS FROM THE MAIN PROGRAM

Program MAIN

AIRFOL	- generate grid
CLUSTR	- cluster grid points near the surface for viscous calculations
METRIC	- compute metrics and Jacobian

Loop:

ROTGRID	- rotate grid to current angle of attack
METRIC	- recompute metrics
SLPS	- perform ADI sweeps
WALLBC	- apply boundary conditions on surface and cut
CPPLOT	- output surface pressure distribution
LOAD	- compute integrated lift, moment, and drag coefficients

The iterative solution loop is designed to execute the ADI sweeps for a number of steps specified in the input data. For restarts of dynamic calculations, the iterative solution loop begins its first cycle by rotating the grid (subroutine ROTGRID) based on the time read from the previously saved solution and the frequency of oscillation. No rotation is required at the initiation of dynamic calculations or for steady-state calculations since the initial conditions set the freestream flow direction at an angle equal to the initial angle of attack. On each subsequent iteration the grid is rotated an incremental amount based on the size of the time step. Following each grid rotation, subroutine METRIC is again called to recompute the metrics since the computational plane is stationary. The actual solution is now accomplished by a call to SLPS, and boundary conditions are applied at each step by WALLBC. The completed solution

may now be used to produce normal program output, at intervals specified in the program. CPPLOT is called to output the surface pressure distribution, and LOAD computes the integrated lift, drag, and moment coefficients. After exiting the loop, various output options may be selected, including the option of saving the current solution for a subsequent restart; creating the plotting files; and writing the complete flow-field solution, including the velocity magnitude, eddy viscosity, and normal distance from the surface at each grid point. The present version of the program is modified to exit at a number of equal time intervals each cycle, as specified within the program. This was done to generate data files for PLOT3D (an Ames Research Center plotting code) at equal phase intervals, and also to save the current solution so it may be written to Cray tapes for possible future use.

B. GRID GENERATION

The call to subroutine AIRFOL initiates the basic grid generation process. AIRFOL begins by reading the airfoil shape input parameters. These include the number of points on the upper and lower surfaces and a flag indicating airfoil symmetry. For symmetric airfoils, only the upper surface coordinates are read, and the lower surface coordinates are defined as (XL, YL) = (XU, −YU). Then the coordinates are scaled with respect to the airfoil chord length, c, and the actual grid generation begins. The sequence of subroutine calls is contained in Table 2.

The first step in the grid generation is to fill in the definition of the airfoil surface. Subroutine SING is called to determine the singular point (x_N, y_N), defined earlier as lying midway between the nose and the origin of the leading-edge radius. SING applies a three-point parabolic curve fit to the leading-edge points to determine the radius of curvature. The angle of the trailing-edge upper surface relative to the leading edge and the average of upper-surface leading- and trailing-edge slopes are also computed for use in determining the wake shape. AIRFOL next calls subroutine TABINT to compute the additional points required. TABINT first performs the mapping to an intermediate plane described by Equation 2.1. The distance required between points to place 97 points on the airfoil surface is calculated. Then the actual determination of the points is performed by a standard polynomial interpolation routine, TAINT. Finally, TABINT maps the smoothed surface back to the physical plane.

TABLE 2
GRID GENERATION SUBROUTINES CALLED

AIRFOL	- called by MAIN
SING	- determine singular point (x_N, y_N)
TABINT	- find additional points on airfoil surface
TAINT	- interpolation routine
WRAP	- calculate stretching in η-direction
CLUSTR	- called by MAIN for viscous flows
STRTCH	- calculate new stretching factor
TAINT	- locate new normal grid point locations

A smooth airfoil shape has now been defined. It remains only to determine the shape of the wake--the "cut" in the physical domain--before constructing the grid. AIRFOL uses the trailing-edge angle computed by SING to calculate a wake shape allowing the flow to leave the trailing edge smoothly. This cut generally corresponds to the tangent of the mean camber line, and for symmetric airfoils, it is just the extended chord line. [Ref. 7: p. 41] It remains at a fixed angle, and the entire grid is rotated along with the airfoil. (See Fig. 2.4)

Construction of the grid is finally completed by subroutine WRAP. The stretching in the normal direction is computed. Then the airfoil and wake are unwrapped using Equation 2.1. This creates a grid consisting of constant-η lines equidistant from the unwrapped surface and constant-ξ lines parallel to the η-axis. This grid is used for inviscid (Euler) calculations. Following the return from WRAP, the last step performed by AIRFOL is to map the entire grid back into the x-y plane, and the coordinate axis is shifted to the quarter-chord point.

If the Reynolds number is greater than zero, program MAIN calls subroutine CLUSTR to construct new constant-η lines. (Setting the Reynolds number to zero or a negative value is a flag for inviscid calculations). For each grid x-coordinate, the old stretching in the normal direction is computed; that is, the physical distance from the surface to each point along constant-ξ lines is calculated. Then subroutine STRTCH

is called to calculate the new stretching. In STRTCH the user-input distance of the first point off the wall and the distance just calculated by CLUSTR to the farthest grid point are used as the inner and outer bounds of the grid. A stretching factor R is desired to give a geometric progression of grid spacings such that, given the distance of the first point off the wall, the ratio between successive spacings is R. This correct stretching factor is calculated by iteration using Newton's method. Once the stretching factor has been determined, CLUSTR calls the interpolation routine TAINT to locate the new grid coordinates.

Construction of the grid in both the physical and computational planes is now complete. The physical grid is easily rotated by subroutine ROTGRID using the relations $x = x \cos(\Delta\alpha) - y \sin(\Delta\alpha)$ and $y = y \cos(\Delta\alpha) + x \sin(\Delta\alpha)$. The only step remaining is the calculation of the transformation metrics and Jacobian. Subroutine METRIC is called by MAIN after construction of the grid and after each rotation. METRIC performs calculations as outlined in the previous chapter under "Numerical Solutions". The grid velocity components are x_τ and y_τ, determined from the angular velocity and grid coordinates in the physical plane (since coordinates were defined with respect to the quarter chord). Central differences are used at interior points and one-sided differences at the boundaries--two-point downstream and three-point elsewhere--to compute x_ξ, x_η, y_ξ, and y_η. This calculation is simplified by the assumption of a unit grid spacing in the computational plane. Finally, the relations given by Equations 2.8 and 2.9 are used to compute the Jacobian and metrics.

C. COMPUTATIONAL STEPS

The basic features of the computational scheme were outlined in the preceding chapter. The organization of subroutines to implement this scheme is presented in Table 3. Program MAIN calls subroutine SLPS to perform the primary calculation steps. SLPS controls the execution of the ADI sweeps, assembles the matrices, collects the damping terms, and calls the matrix solvers. The value of the implicit damping coefficient is set in relation to the explicit coefficient. In the present implementation $\varepsilon_I = 20\varepsilon_E$. If the reduced frequency is less than .001, a local time-step option is used to accelerate convergence to a steady-state solution by basing the time step on the local flow conditions. The calculations begin by calling subroutine DISSIP to compute the explicit dissipation terms described in the last chapter and add them to the right side of the governing equation (Eqn. 2.20). In the ξ-direction DISSIP uses a blend of second-

and fourth-order terms. A switching function is computed, based on the second derivative of pressure. This function causes the fourth-order terms to be set to zero when the pressure gradient exceeds a value specified within the program. Otherwise the second-order terms are set to zero. The large pressure gradients experienced in the vicinity of shocks are not expected in the η-direction, so only fourth-order terms are added. The arrays DQ1-DQ4 contain the combined dissipation terms.

TABLE 3
SUBROUTINES USED IN COMPUTATIONAL STEPS

SLPS	- called by MAIN
DISSIP	- compute explicit dissipation terms
RESI	- compute inviscid right-hand (explicit) terms
STRESS	- compute viscous right-hand terms
EDDY	- determine viscosity coefficient
AMAT1	- compute Jacobian matrix [A]
MATRX1	- invert assembled matrix in the ξ-direction
AMAT2	- compute Jacobian matrix [B]
MATRX2	- invert assembled matrix in the η-direction
WALLBC	- called by MAIN to apply surface boundary conditions

Subroutine SLPS next calls RESI to compute the inviscid right-hand terms at the known time level (Eqn. 2.18). Working first with the ξ-derivative flux terms, RESI begins by calculating the contravariant velocity component U, in the computational plane (Eqn. 2.13). The vector $\mathbf{F}^*$ (Eqn. 2.14) is then formed using the contravariant velocity. Then $-\Delta t[\partial_\xi(\mathbf{F}^*)^n]$ is computed using standard central differences and added to the arrays DQ1-DQ4. These steps are repeated in the η-direction, using the contravariant velocity component V, and $-\Delta t[\partial_\eta(\mathbf{G}^*)^n]$ is added to the DQ arrays.

For viscous flows, subroutine STRESS is called next to compute the viscous terms and add them to the right-hand side of the equation. The first step in STRESS is a call to EDDY to compute the total (turbulent and laminar) viscosity. (EDDY is

not called on the first ten steps of the initial dynamic run to allow the solution to settle a bit). The eddy viscosity routine begins by initializing the viscosity throughout the field to the laminar value calculated from the Mach number and Reynolds number. If laminar calculations are desired, the subroutine should be exited at this point. A completely turbulent flow field is assumed, so no transition point must be determined. The subroutine works with one grid ξ-location at a time. The calculations performed were described in the last chapter under "Turbulence Model". The variables defined in that section are initialized to low values which will be reset as the flow field is scanned in a normal direction from the surface. Beginning at the airfoil boundary, the derivatives u_η, v_η are calculated by one-sided differences, and the metrics and Jacobian are again computed as described previously. These values are all local to subroutine EDDY. The no-slip condition is now applied to determine the viscous stress, τ_w, and skin friction. The friction value, CF, is one of the selectable normal output values from MAIN. The Van Driest damping factor, y^+/A^+, is calculated and the subroutine begins scanning the grid points outward from the surface. The values u_ξ, v_ξ, u_η, v_η; the metrics and Jacobian; and U_{dif}, y, and F are calculated, and the values of F_{max} and y_{max} are found (Eqns. 2.27 and 2.28). If the angle of attack is less than the user-specified value ALFAI and is increasing or constant, the modified turbulence model is used and F_{max} is the value of F where F • y is maximum. Above ALFAI on the upstroke and on the entire downstroke, the original Baldwin-Lomax model is used and F_{max} is the maximum value of F. Then the length scale, ℓ, is found using the Van Driest function, and the inner-layer viscosity is computed (Eqn. 2.24). Next, the outer-layer formula previously described is used to find a value for viscosity at each normal grid location; and the point where the inner- and outer-layer values first cross is found. The final step performed by EDDY is to add the laminar viscosity to the appropriate inner- or outer-layer turbulent viscosity.

Now that the viscosity has been calculated, subroutine STRESS can carry out evaluation of $(\partial_\xi \mathbf{R}^* + \partial_\eta \mathbf{S}^*)$ at the known time level, where $\mathbf{R}^*$ and $\mathbf{S}^*$ are, as given by Equation 2.12, combinations of the vector $\mathbf{R}$ and $\mathbf{S}$. First the derivatives in the ξ-direction are computed by central differences. A more efficient discretization of the spatial derivatives of viscous terms is realized by evaluation at half points, rather than at the nodal points. This reduces the number of nodal points involved from five to three. [Ref. 7: p. 26] Therefore, STRESS computes derivatives, metrics, Jacobian, and an average viscosity at the half points. These values are used to determine τ_{xx}, τ_{xy},

τ_{yy}, R_4, and S_4. In the present implementation R_4 and S_4, the viscous work (energy dissipation) and conduction (diffusion) terms, are set equal to zero. Finally the differenced value of $\partial_\xi \mathbf{R}^*$ is added to the DQ1-DQ4 arrays. The same steps are then followed for the η-derivative terms.

Subroutine SLPS has now assembled in the arrays DQ1-DQ4, the elements of the right-hand side of Equation 2.20 given by Equation 2.18, with explicit damping added but without the factor Δt. Next, the subroutine AMAT1 is called to compute the Jacobian matrix $[A] = \partial_q \mathbf{F}^*$, and the left-hand side of Equation 2.20 is then assembled. The second-order implicit damping terms are computed and added to the left side. Finally, the right-hand side (DQ arrays) is multiplied by Δt and stored in matrix GG, giving the final form of the right-hand side of Equation 2.20. A standard matrix inversion routine, MATRX1, obtains the solution to Equation 2.20, and Step 1 of the Beam-Warming algorithm, the ξ-sweep, is now complete. The solution, corresponding to $\Delta \mathbf{q}^{**}$ in Equation 2.20, is now stored in the DQ1-DQ4 arrays to become the right-hand side of Equation 2.21. The steps used in the ξ-sweep are now repeated for the η-direction. AMAT2 computes the Jacobian matrix $[B] = \partial_q \mathbf{G}^*$, and the left-hand side of Equation 2.21 is assembled with implicit damping again added. MATRX2 is called to obtain the solution $\{\Delta \mathbf{q}^*\}^{n+1}$ and complete Step 2 of the Beam-Warming algorithm. Step 3 is accomplished by updating the flow variables (Eqn. 2.22). Values at the boundaries are not updated. The outer boundaries remain at undisturbed freestream values making a specific step to apply boundary conditions unnecessary. This completes the solution procedure, but as a monitoring feature, the density elements of $\Delta \mathbf{q}^*$ (now stored in DQ1) are scanned for the largest value. This value, along with its grid coordinates, is output at specified intervals, giving an indication of the most rapidly changing area of the flow field and the magnitude of the density change there.

A solution has now been obtained by program MAIN. It remains only to apply the surface boundary conditions. This is accomplished by subroutine WALLBC. Boundary conditions were discussed in the last chapter, and the implementation is straightforward. Conditions along the cut are averaged from above and below. The contravariant velocity component U at the two points nearest to the surface is used to determine U at the surface for Euler calculations; for viscous flows U = 0 at the surface. The appearance of the calculation steps is complicated slightly by use of a scheme that is applicable to both viscous and inviscid flows. Surface flow properties are finally obtained by extrapolating from interior points.

IV. RESULTS AND DISCUSSION

Nine simulations are described in this chapter. Several variations were tried to investigate the effects of varying input parameters, some of which are mentioned without detailed presentation of results. The intention was to obtain complete solutions under a range of conditions of dynamic effects, Reynolds number, Mach number, and turbulence model, whether or not such solutions are the most accurate obtainable under those conditions. It was hoped thereby to gain some understanding of the most effective employment of the code, its limitations, and possibilities for improving its performance. Table 4 lists the input parameters for the simulations. The calculations were grouped into the following areas:

1. Steady-state calculations at and beyond maximum lift.
2. Comparison with deep stall experimental data, using the original Baldwin-Lomax turbulence model and modification.
3. Comparison to experimental data for fully attached flow using both turbulence model versions.
4. Low Reynolds number predictions at Mach numbers of .3 and .5.

All of the runs were made with the implicit damping coefficient ε_I at a value of twenty times the explicit value ε_E (input as "WW"). The distance of the first point off the solid boundary was set to 0.00005 for all but the steady-state calculations beyond maximum lift.

Three basic types of output were obtained:

1. The flow-field solution values ρ, ρu, ρv, and e and the corresponding grid coordinates at twelve equal time intervals for each cycle, for use in the plotting program PLOT3D by Pieter Buning of NASA Ames. This routine was used to produce contour plots of pressure, density, Mach number, and stream function.
2. Surface pressure coefficients at 96 equal time intervals for each cycle. This data was used in the plotting routine CARPET, written by Rosalie Lefkowitz of Sterling Software, to produce pressure distribution carpet plots utilizing the DISSPLA graphics library.
3. The normal text output file containing, for 96 points each cycle, the integrated lift, drag, and moment coefficients; pressure distributions; the values and locations of the largest solution increments after every ten time steps; and skin friction values. These coefficients were plotted using the DISSPLA plotting package QPLOT available on the NASA Ames VAX computers.

The use of equal time intervals for output clusters the data near maximum and minimum angles of attack, where interest is often greatest. The solution was also saved on Cray tapes, at twelve points each cycle, for future program restarts, so a more

thorough investigation can be conducted of any interval with minimal use of computer time.

TABLE 4
INPUT CONDITIONS FOR COMPUTER SIMULATIONS

1. $\alpha = 12^\circ$, $M_\infty = .3$, Re = 1,000,000
   Baldwin-Lomax Turbulence Model

2. $\alpha = 14^\circ$, $M_\infty = .3$, Re = 1,000,000
   Baldwin-Lomax Turbulence Model

3. $\alpha = 13.5^\circ$, $M_\infty = .301$, Re = 3,910,000
   Modified Turbulence Model

4. $\alpha_0 = 15^\circ$, $\alpha_1 = 10^\circ$, $M_\infty = .283$, Re = 3,450,000
   Baldwin-Lomax Turbulence Model

5. $\alpha_0 = 15^\circ$, $\alpha_1 = 10^\circ$, $M_\infty = .283$, Re = 3,450,000
   Modified Turbulence Model

6. $\alpha_0 = 8^\circ$, $\alpha_1 = 10^\circ$, $M_\infty = .184$, Re = 2,450,000
   Baldwin-Lomax Turbulence Model

7. $\alpha_0 = 8^\circ$, $\alpha_1 = 10^\circ$, $M_\infty = .184$, Re = 2,450,000
   Modified Turbulence Model

8. $\alpha_0 = 15^\circ$, $\alpha_1 = 10^\circ$, $M_\infty = .284$, Re = 345,000
   Baldwin-Lomax Turbulence Model

9. $\alpha_0 = 15^\circ$, $\alpha_1 = 10^\circ$, $M_\infty = .5$, Re = 345,000
   Baldwin-Lomax Turbulence Model

A. STEADY-STATE CALCULATIONS

Before beginning dynamic simulations, the performance of the program in constant angle of attack calculations was explored. A verification series of runs at increasing angles of attack was made using the conditions reported by Sankar in Reference 6. The results agree with Sankar's and are not reproduced here. The most notable discrepancy from experimental results is the underprediction of maximum lift using the Baldwin-Lomax turbulence model, with separation occurring between one and two degrees below experimental values. The local time-step option, incorporated since Sankar's results were published, produced converged solutions in fewer than 1500 steps at even the higher angles of attack. In addition to the integrated coefficients and surface pressure distribution reported by Sankar, the flow field was graphically analyzed. Figure 4.1 contains density, pressure, Mach number, and stream function plots at 12 degrees angle of attack.

Next the behavior of the solution above static stall angles was investigated. The original Baldwin-Lomax model was used with a Mach number of .3 and a Reynolds number of one million at 14 degrees angle of attack. The time-accurate mode was invoked by using a reduced frequency of .002 with zero amplitude of oscillation. The step size was .005 and the distance to the first point off the wall was .0001. Figure 4.2 presents contour plots of density and pressure coefficient after 5100, 5610, and 6120 steps. Based on the Mach number and time step, this should represent a total movement of three chord lengths of a particle in the freestream. The vortex, however, remains on the airfoil and gradually diminishes in intensity. The integrated lift coefficient steadily decreased throughout the run to a value of less than 0.6.

The modified turbulence model was applied to the conditions of frame 4019 of Reference 8 (volume 3, page 46) at the experimentally observed maximum lift angle of attack of 13.5 degrees. The modified turbulence model provides excellent results. For a Reynolds number of 3.91 million and Mach number of .301, the local time-step option was used with an explicit viscosity input of WW = 2. After 1500 time steps the experimental maximum lift coefficient of 1.4 was obtained, and the peak pressure coefficient of 8.1 appears to be very close to the experimental value. Moment and drag coefficient values also seem reasonable. The theoretical pressure distribution at maximum lift is shown in Figure 4.3, compared to the experimental distribution taken from Reference 8.

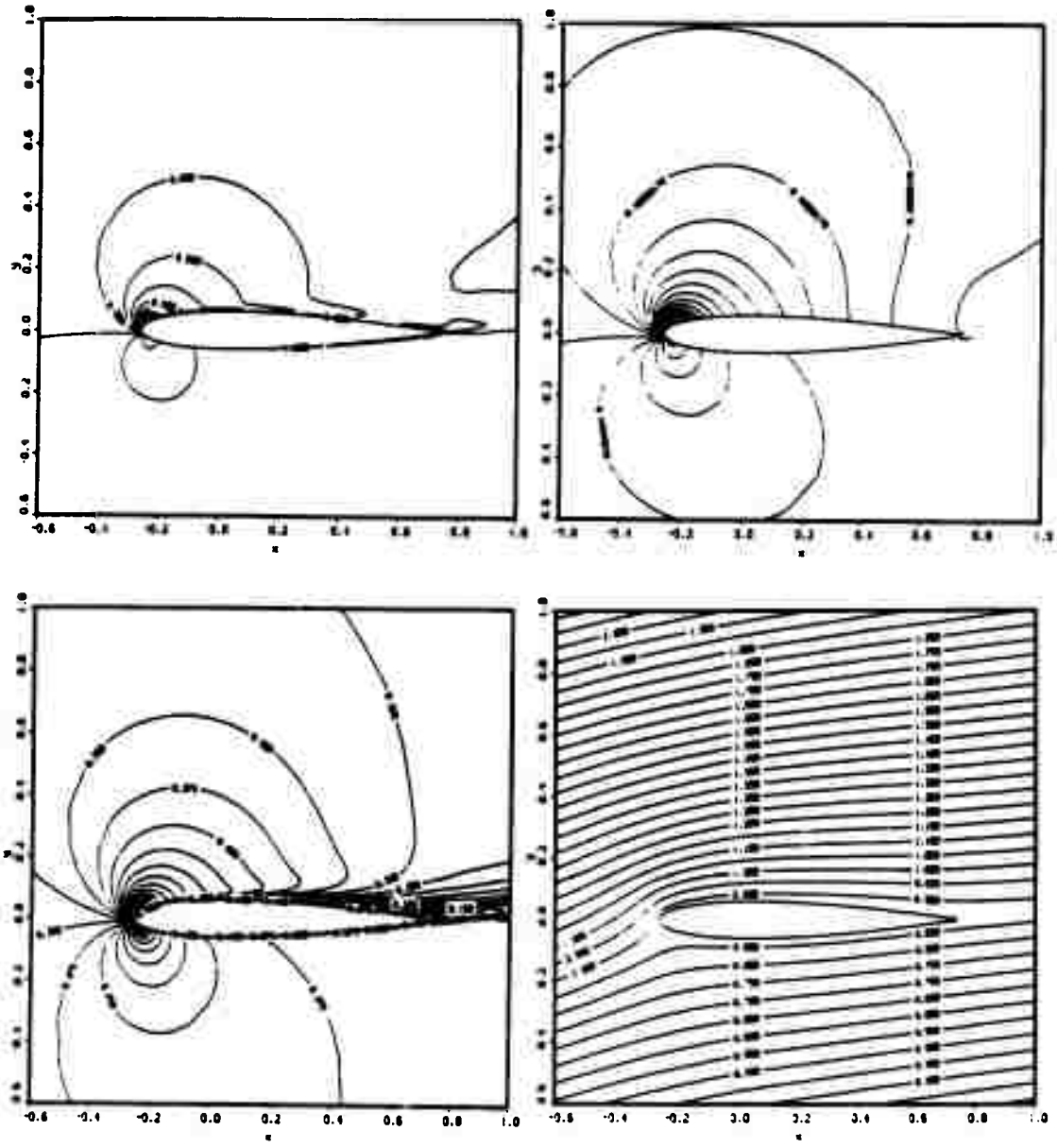


Figure 4.1 Steady-State Attached Flow

$M_\infty = .3$, $Re = 1.00 \times 10^6$, $\alpha = 12°$

Top--Density (l), Pressure (r) Bottom--Mach No. (l), Stream Function (r).

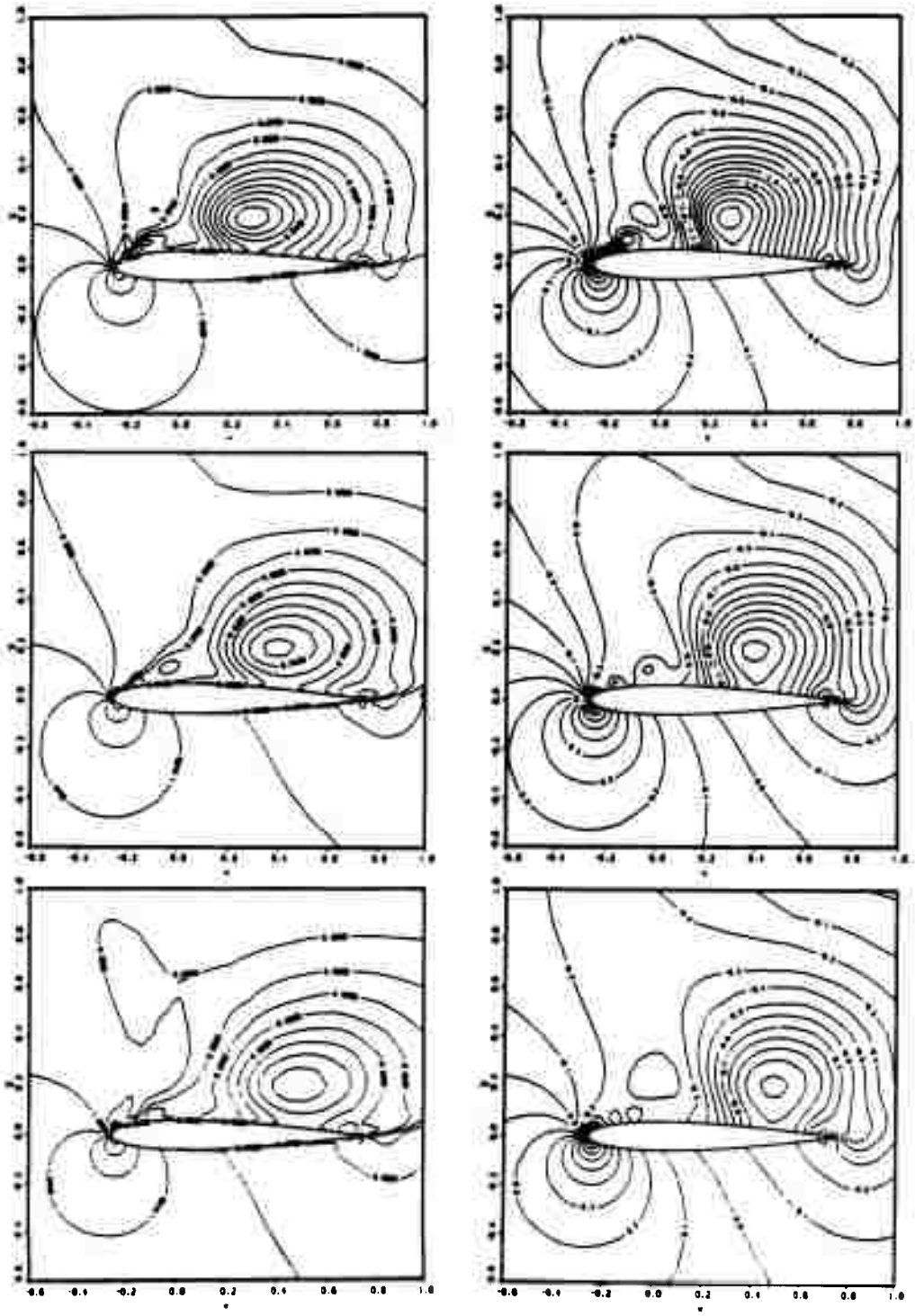


Figure 4.2 Steady-State Separated Flow

$M_\infty = .3$, $Re = 1.00 \times 10^6$, $\alpha = 14°$

Density (l) and Pressure (r) at 5100, 5610, and 6120 time steps.

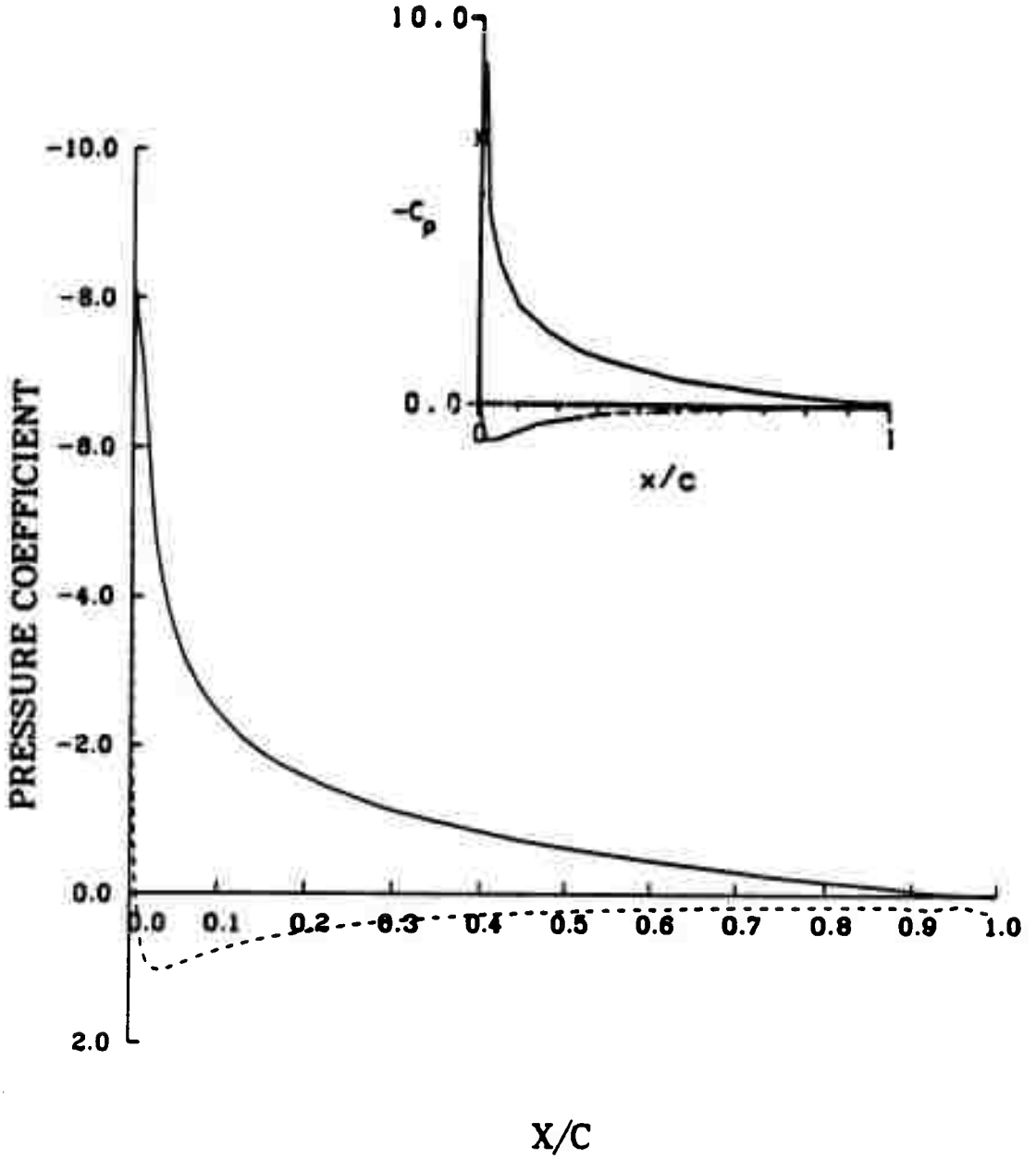


Figure 4.3 Steady-State Pressure Distribution at Maximum Lift
$M_\infty = .301$, $Re = 3.91 \times 10^6$, $\alpha = 13.5°$
Inset--Experimental Data (Ref. 8).

B. COMPARISON WITH DEEP STALL EXPERIMENTAL DATA

The first dynamic case studied was under the experimental conditions of frame 9218 of Reference 8 (volume 3, page 146). These are the same conditions as Sankar used for comparison in Reference 6. Sankar reported only the integrated lift, moment, and drag coefficients, however, and here the surface pressure distributions and the flow field about the airfoil were investigated. Present calculations also used the exact Mach number and full Reynolds number, and runs were made with both the original Baldwin-Lomax model and the modification. This experimental case closely corresponds to the model dynamic stall (Figure 1.1), showing the formation of a strong vortex, an increase in maximum lift, rapid variations in pitching moment, and the typical lift versus angle of attack hysteresis loop. The program was run under the following conditions:

$Re = 3.45 \times 10^6$

$M_\infty = .283$

$\alpha_0 = 15^\circ$

$\alpha_1 = 10^\circ$

$k = .151$

For the first run the original Baldwin-Lomax turbulence model was used with an explicit damping input of 5. The time step was a constant .005 throughout, and data was saved beginning at the mean angle on the second oscillation cycle. Figure 4.4 shows the lift and pitching moment coefficients plotted against angle of attack for both theoretical and experimental data. Figure 4.5, a plot of the pressure drag coefficient, is included for completeness; drag plots are not presented for the remaining cases, since they provided no additional information for this study. Figure 4.6 contains a carpet plot of the surface pressure coefficients for theoretical data and a plot of experimental data taken from Reference 8. Figures 4.7 to 4.14 are flow-field plots made at twelve equal time intervals during the cycle. The results show excellent qualitative agreement with experimental data throughout the cycle. The moment coefficient is consistently low during the upstroke, but reasonable quantitative agreement of the lift coefficient is obtained until near 18 degrees. Just as in the steady-state case, the Baldwin-Lomax model gives an early prediction of flow separation. The maximum lift does not reach as high a value as experimental results, since separation occurs at a lower angle of attack in the theoretical results; as is the case for the experimental data, the lift continues to increase after the pressure distribution, indicated in the carpet plot, begins to break down. At the beginning of the downstroke, as the vortex is shed off the

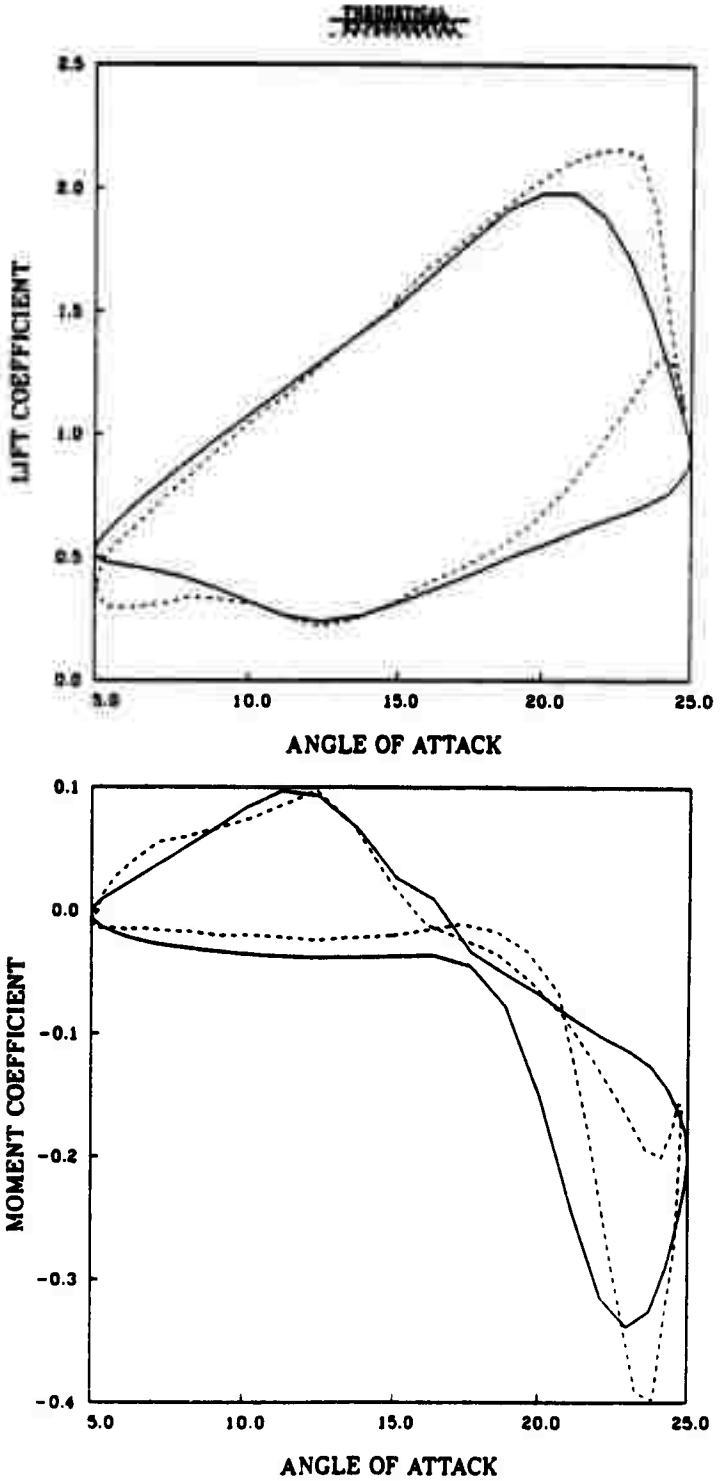


Figure 4.4 Lift and Pitching Moment Coefficients--Baldwin-Lomax Model $M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15^\circ - 10^\circ \cos(.3t)$.

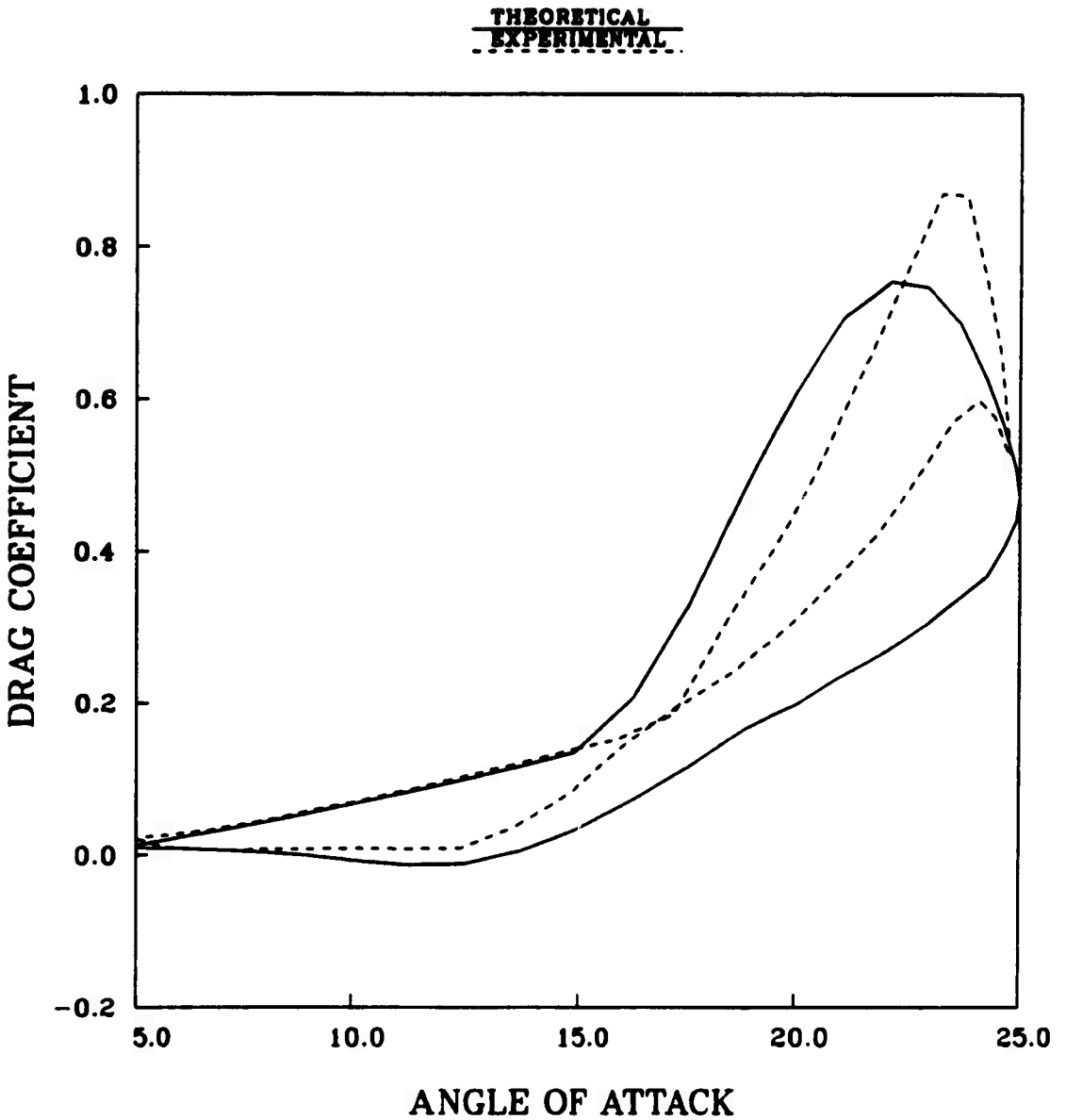


Figure 4.5 Pressure Drag Coefficient--Baldwin-Lomax Model
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15^\circ - 10^\circ \cos(.3t)$.

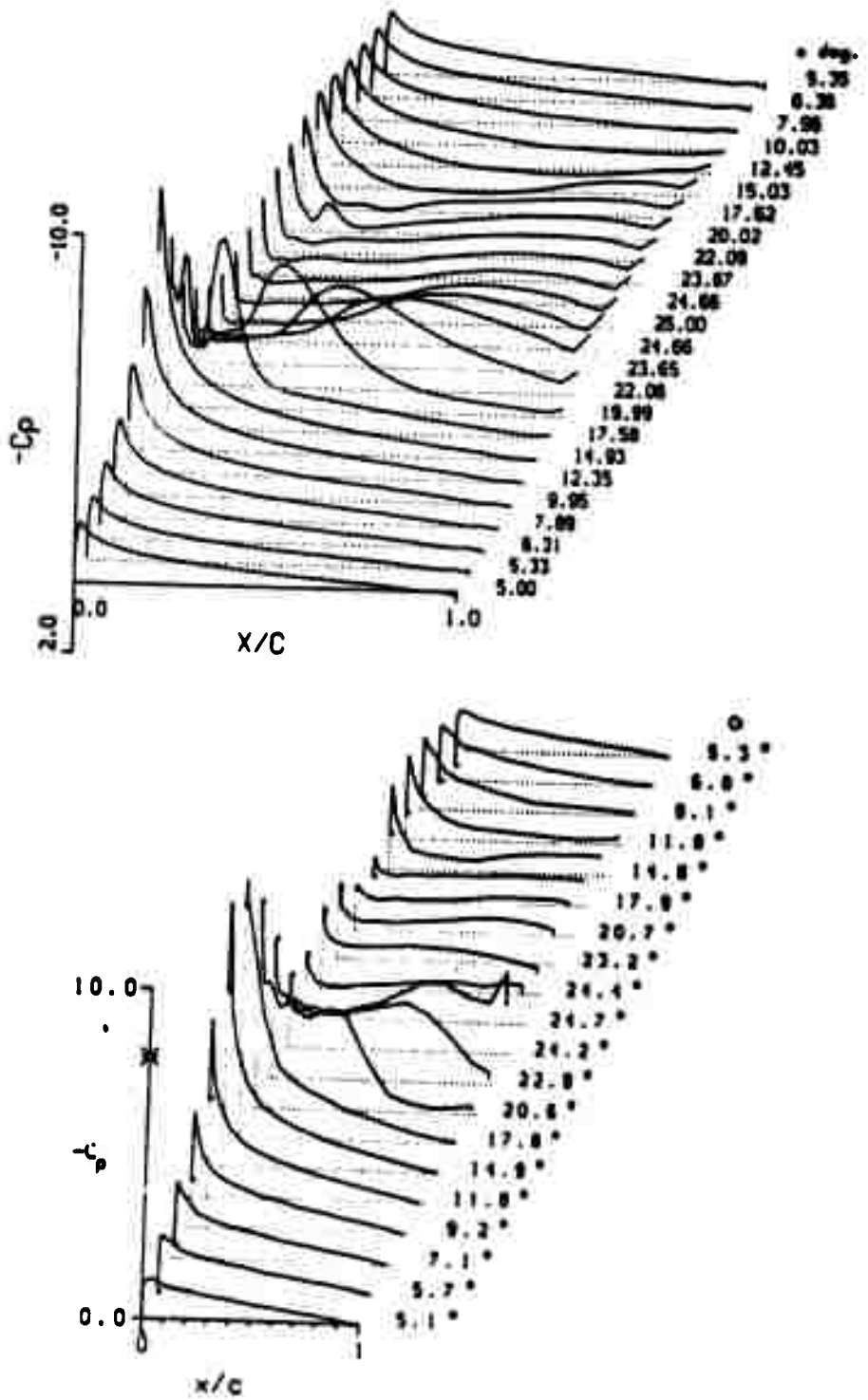


Figure 4.6 Surface Pressure Coefficient Carpet Plots--Baldwin-Lomax Model
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15° - 10° \cos(.3t)$,
Top--theoretical, Bottom--experimental (Ref. 8).

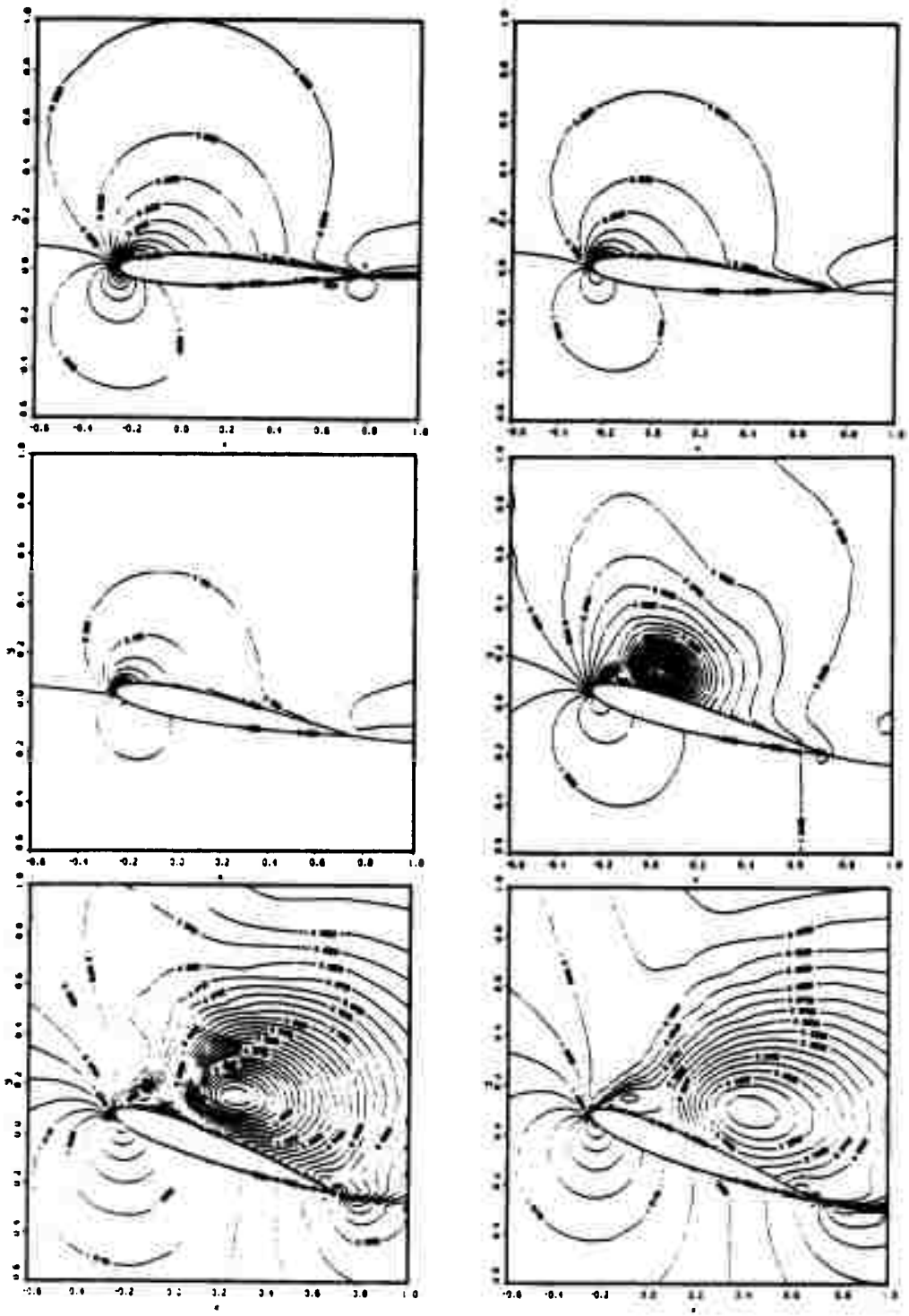


Figure 4.7 Density Contour Plots, Upstroke--Baldwin-Lomax Model
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15° - 10° \cos(.3t)$
6.31, 9.95, 14.93, 19.99, 23.65, 25 degrees.

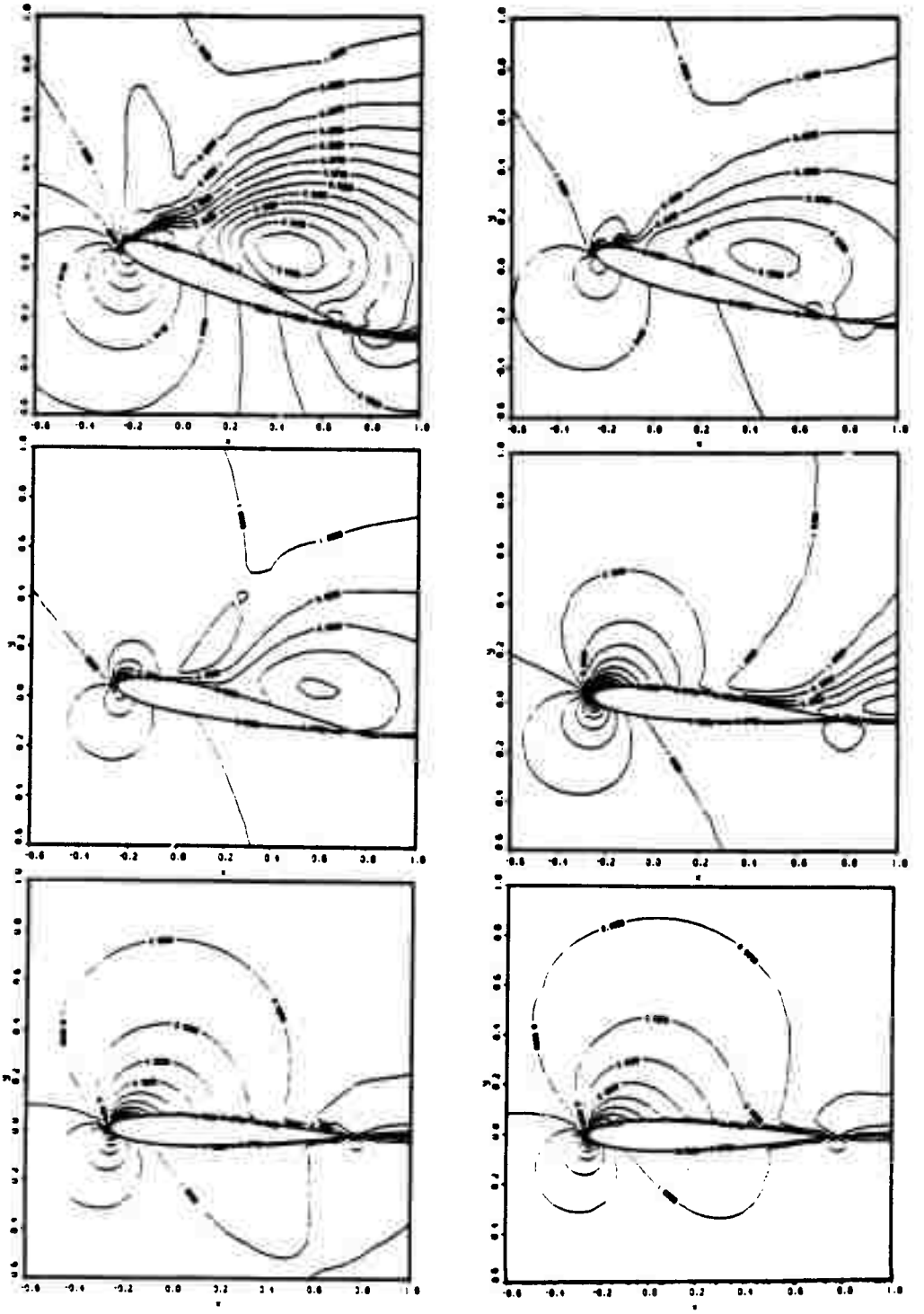


Figure 4.8 Density Contour Plots, Downstroke--Baldwin-Lomax Model
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15^\circ - 10^\circ \cos(.3t)$
23.67, 20.02, 15.03, 10.03, 6.36, 5 degrees.

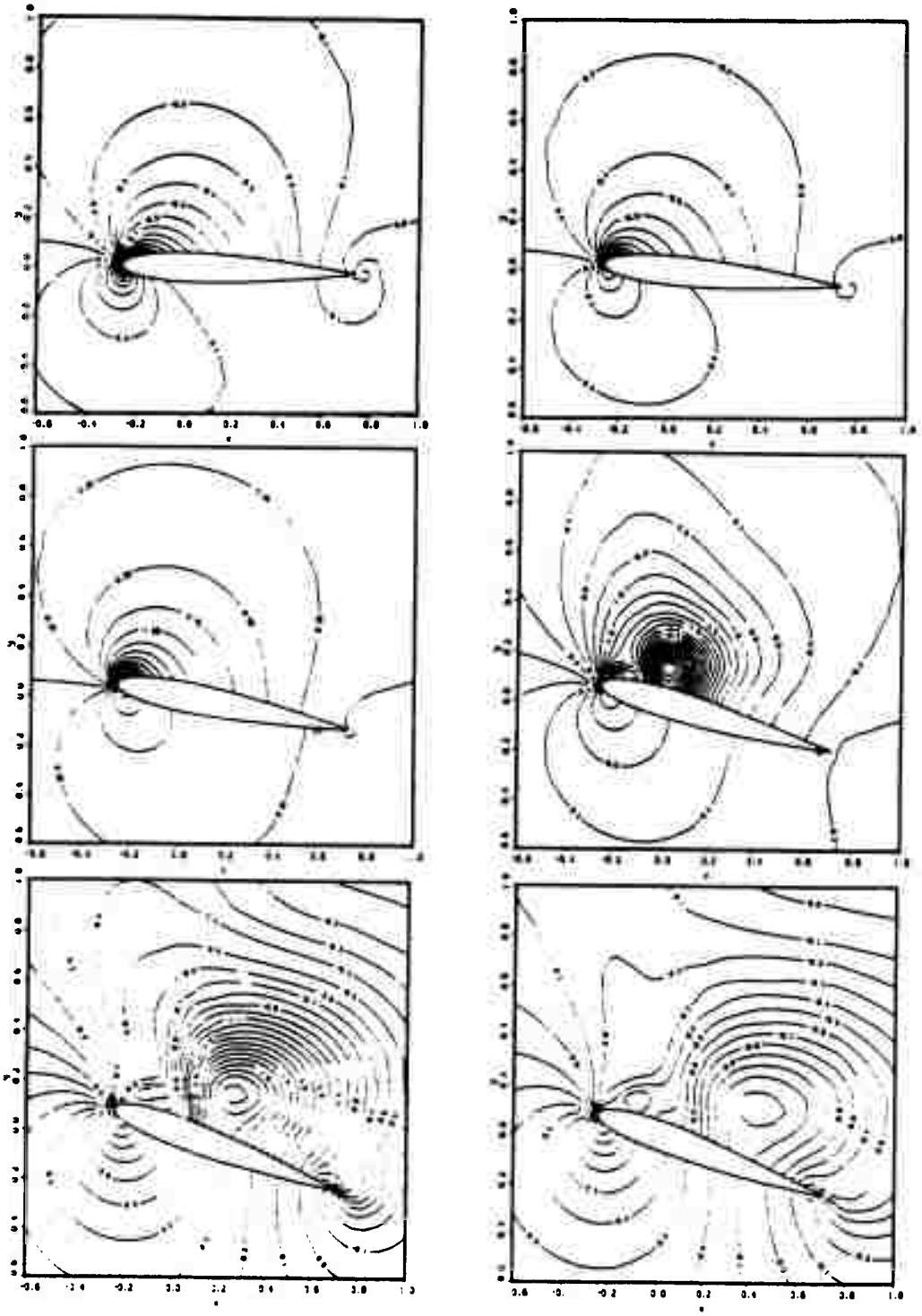


Figure 4.9 Pressure Contour Plots, Upstroke--Baldwin-Lomax Model
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15° - 10° \cos(.3t)$
6.31, 9.95, 14.93, 19.99, 23.65, 25 degrees.

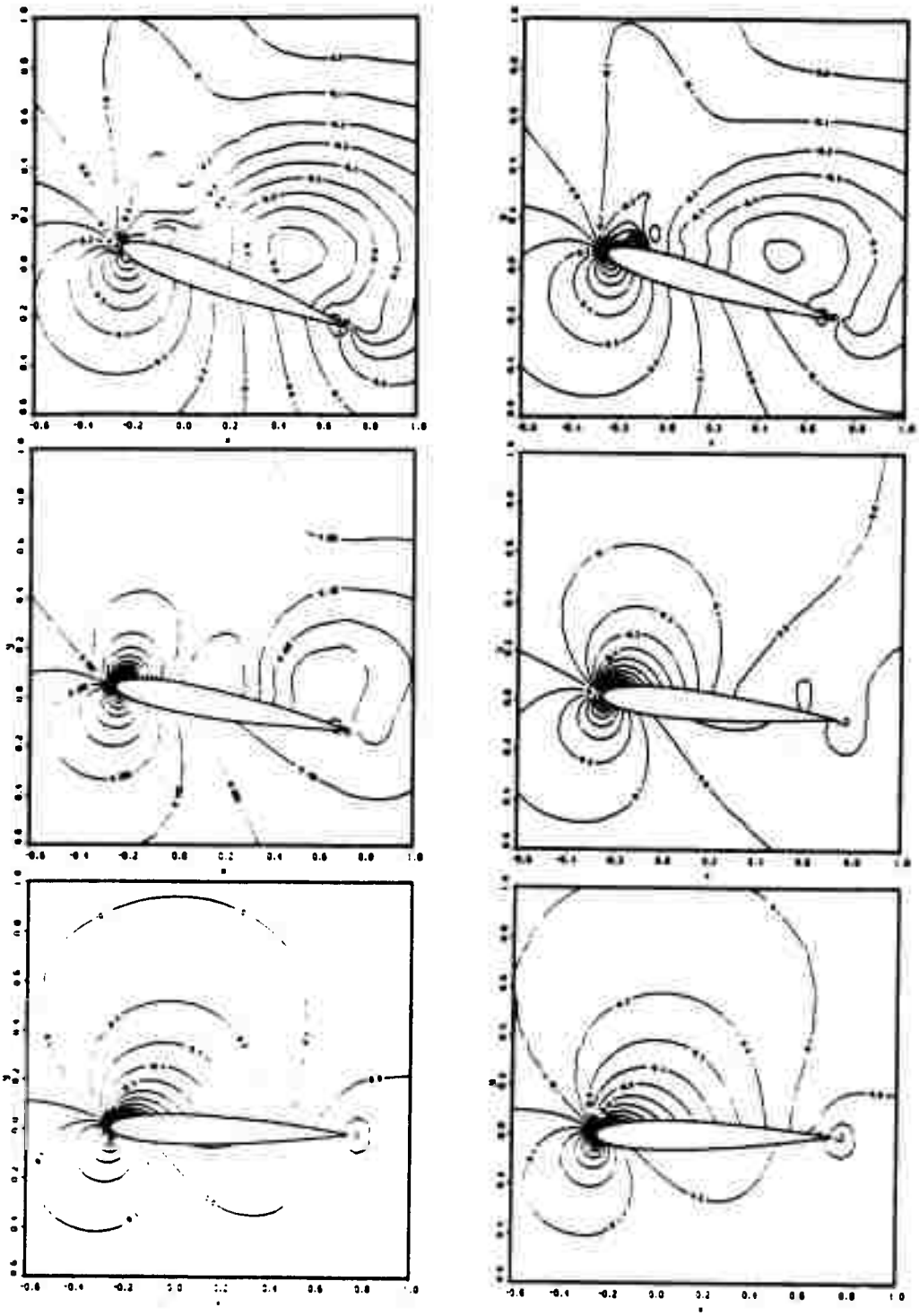


Figure 4.10 Pressure Contour Plots, Downstroke--Baldwin-Lomax Model
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15° - 10° \cos(.3t)$
23.67, 20.02, 15.03, 10.03, 6.36, 5 degrees.

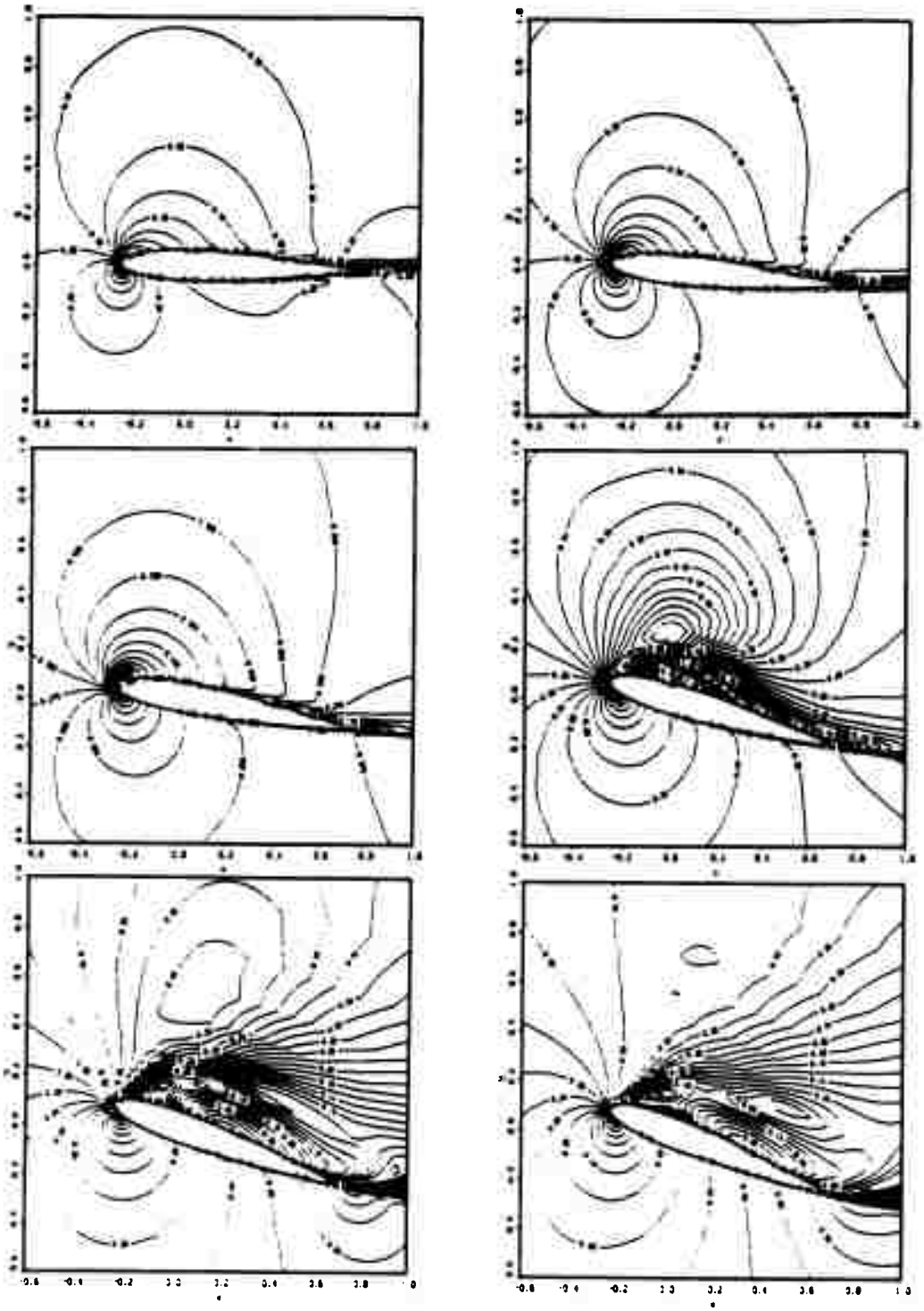


Figure 4.11 Mach Number Contour Plots, Upstroke--Baldwin-Lomax Model
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15^\circ - 10^\circ \cos(.3t)$
6.31, 9.95, 14.93, 19.99, 23.65, 25 degrees.

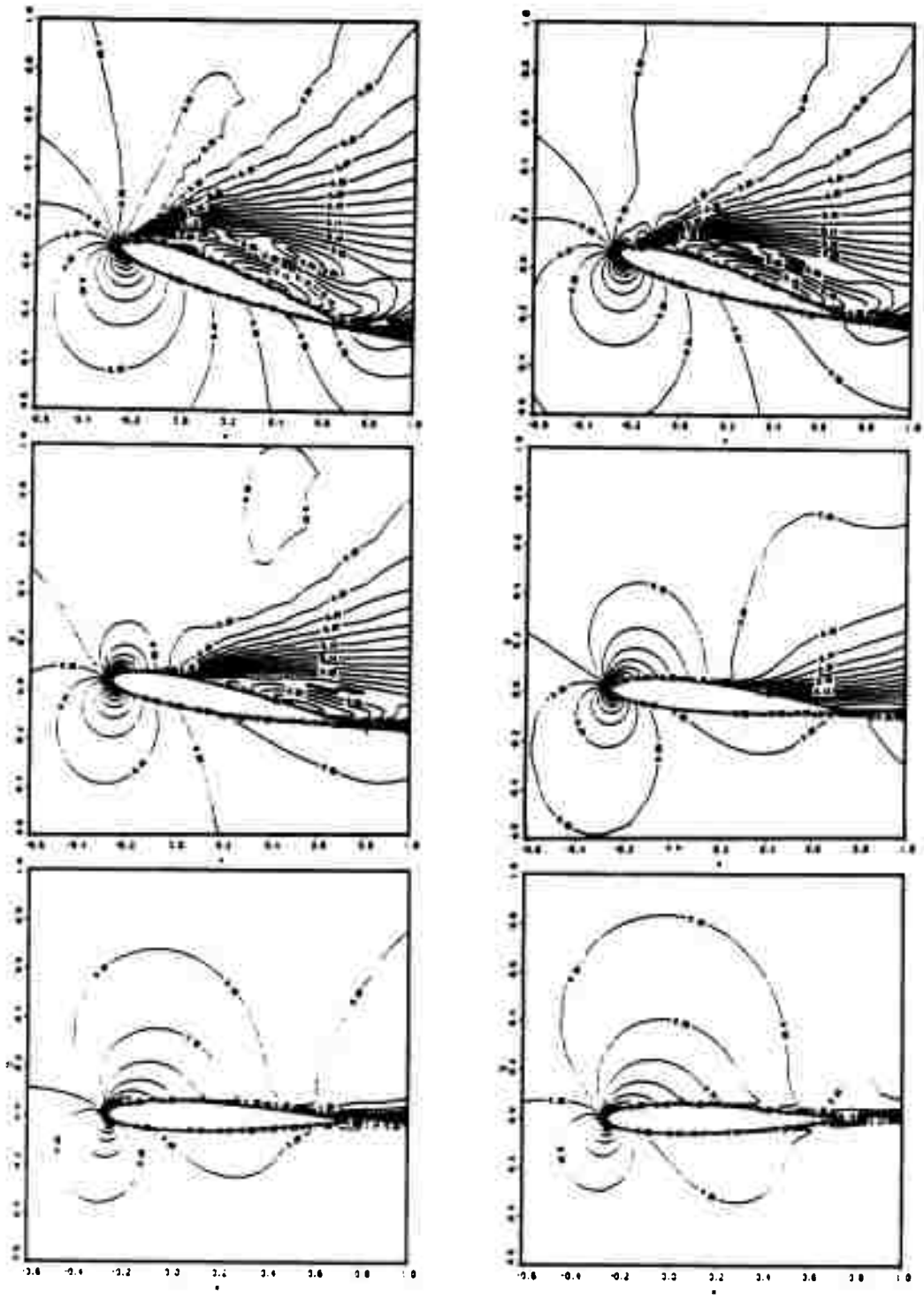


Figure 4.12 Mach Number Contour Plots, Downstroke--Baldwin-Lomax Model
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15° - 10° \cos(.3t)$
23.67, 20.02, 15.03, 10.03, 6.36, 5 degrees.

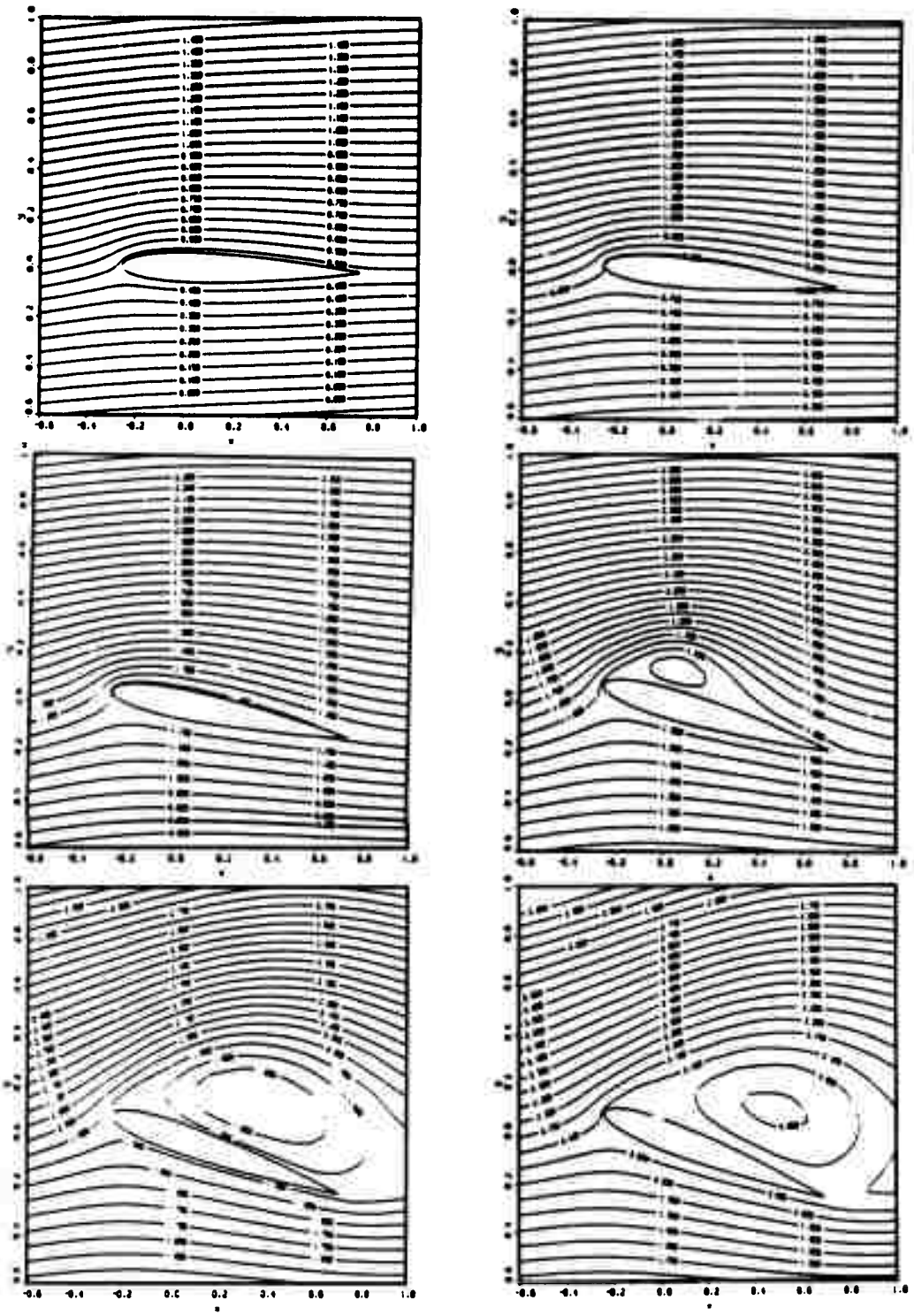


Figure 4.13 Stream Function Plots, Upstroke--Baldwin-Lomax Model
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15° - 10° \cos(.3t)$
6.31, 9.95, 14.93, 19.99, 23.65, 25 degrees.

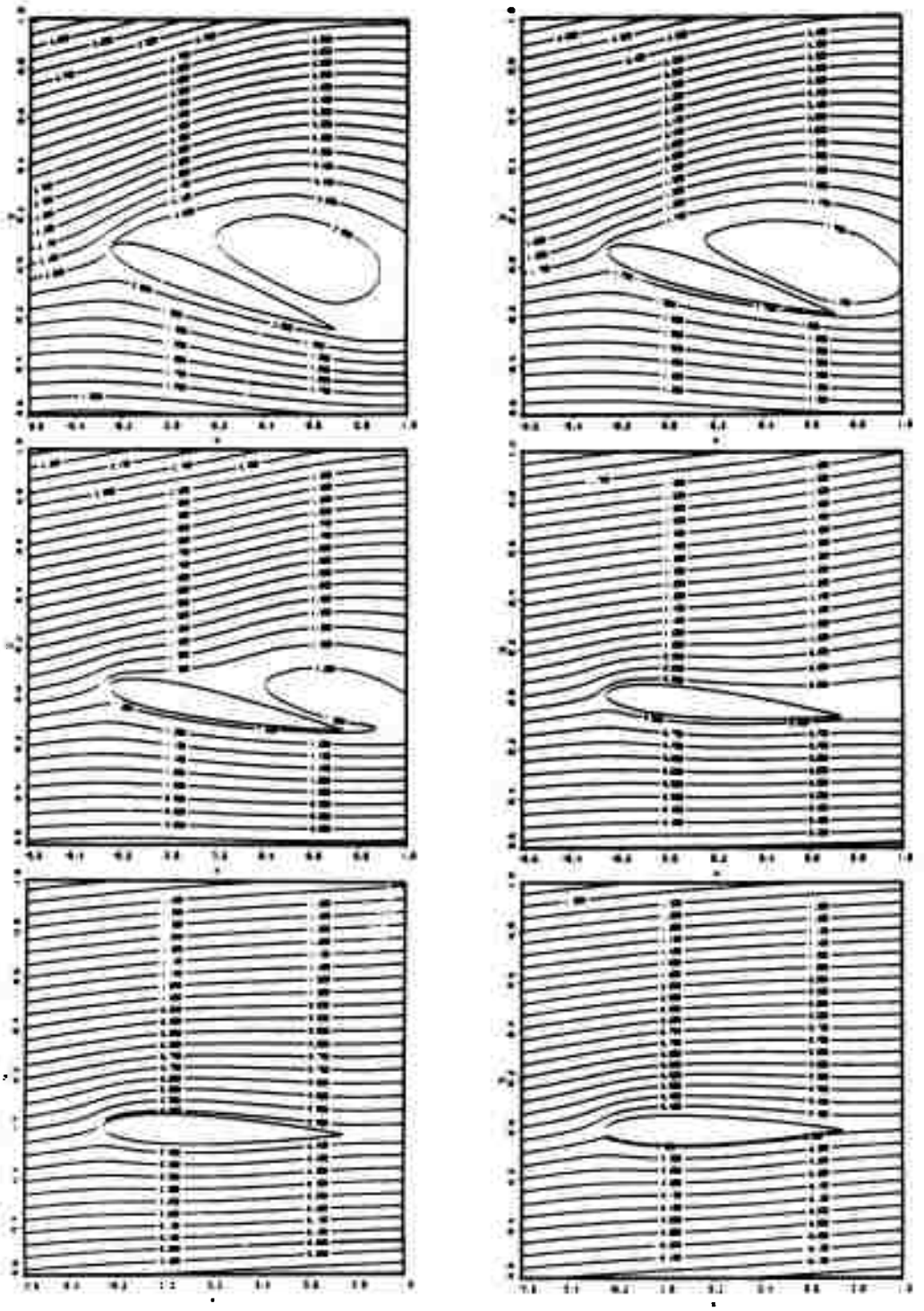


Figure 4.14 Stream Function Plots, Downstroke--Baldwin-Lomax Model
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15° - 10° \cos(.3t)$
23.67, 20.02, 15.03, 10.03, 6.36, 5 degrees.

trailing edge, there is a jump in all the experimental coefficients. This is not duplicated in the theoretical data. The flow-field plots reveal the reason for this difference, as the vortex is not shed immediately but remains on the trailing edge and is dissipated during the downstroke. The absence of strong leading-edge peaks in the carpet plot, until the upstroke, shows the effect of this discrepancy on the pressure distribution. The recovery from separation occurs gradually as the vortex dissipates. Skin friction values are negative over the upper surface until the downstroke begins, and full recovery is not observed until completion of the downstroke.

A run was made under these same experimental conditions but using the modified turbulence model on the upstroke. It was intended to turn off the modified model above 19 degrees, as recommended by Sankar, but due to an error in the code, it was used throughout the upstroke. On the first attempted run, execution terminated as the angle approached 19 degrees. The values of Δq^* had begun growing rapidly near the leading edge, and when a negative value was computed for the density, a math error occurred causing the computer program to terminate abnormally. The surface pressure distribution had been showing unusually high leading-edge peaks, but the integrated coefficient values were close to experimental values when termination occurred. Reducing the time step in half to 0.0025 at 15 degrees allowed the solution to proceed to above 23 degrees, but execution was then terminated in the same manner. The calculations were restarted at this time using a time step of .005 and distance of the first point off the wall of .00005, but this time with the viscosity input, WW = 10. Reduced, but very strong leading-edge suction peaks still appeared, the integrated coefficients were not as large, and a completed solution was obtained without reducing the time step. Data was again saved after one and one-fourth oscillation cycles.

The run using the modified turbulence model shows marked differences from the run with the Baldwin-Lomax model. The lift, drag, and moment coefficients are plotted along with experimental values in Figure 4.15 Now the rapid drop in lift is delayed too long--until the maximum angle is passed. Maximum lift is still underpredicted, but the higher viscosity value might be expected to affect accuracy. The moment coefficient is again lower than experimental during the upstroke until the vortex movement that is not predicted by the code. Note, however, the agreement with experiment once the "theoretical" vortex has been generated. The carpet plot, Figure 4.16, reveals interesting differences from Figure 4.6. Until past the mean angle of 15

degrees, the pressure distributions are nearly identical. In fact, the Baldwin-Lomax model has a slightly stronger suction peak (−7.7 to −7.5 for the modified model). After this point, however, the Baldwin-Lomax model shows the effect of the premature flow separation while the suction peaks for the modified model grow abnormally high. Note the different scales used in the experimental and theoretical carpet plots. Although the integrated lift coefficient begins to drop slightly nearing the maximum angle, the pressure distribution is still apparently undisturbed at 25 degrees, with some weakening of the suction peak. The situation changes abruptly as the downstroke begins and the modified turbulence model is replaced by the original Baldwin-Lomax model. The suction peak finally breaks down, and by 17.6 degrees on the downstroke, the distribution is again very similar to that with the Baldwin-Lomax model. During the upstroke, the first negative surface friction values appear near the trailing edge at 21.6 degrees, and by 22.9 degrees, the negative values extend over the upper surface. This is the only clear indication of the residual stresses that are so abruptly relieved when the airfoil pitches down.

The flow-field plots are similar in many respects to those for the Baldwin-Lomax model except for the timing of the vortex formation. Figures 4.17 and 4.18 are flow-field plots at the maximum angle and at 23.67 degrees on the downstroke. At the maximum angle there is still no strong vortex formation, although a narrow recirculating region is indicated near the trailing edge. The rapid development and shedding of the vortex occurs during the downstroke. By 23.62 degrees, the vortex has progressed well along the upper surface. The sequence of events is reasonable, although the timing is not.

The program was corrected and run again under the same experimental conditions with two approaches to the problem of maintaining stability at the higher angles. First, the solution was marched from the mean angle (using the solution saved in the previous run at 15 degrees) to the maximum angle with the time step left at a constant .005 and the viscosity increased to 10. Then the run was repeated with the viscosity input left at 5 and the time step decreased to .0025 (as had been attempted earlier with the faulty turbulence model). This latter run was continued into the downstroke using a time step of .005, and the integrated lift and pitching moment coefficients are plotted in Figure 4.19. The results are improved over those with the Baldwin-Lomax model (Fig. 4.4). The maximum lift of 2.1 matches experiment and the lift curve slope is sustained longer into the upstroke. The negative moment peak is

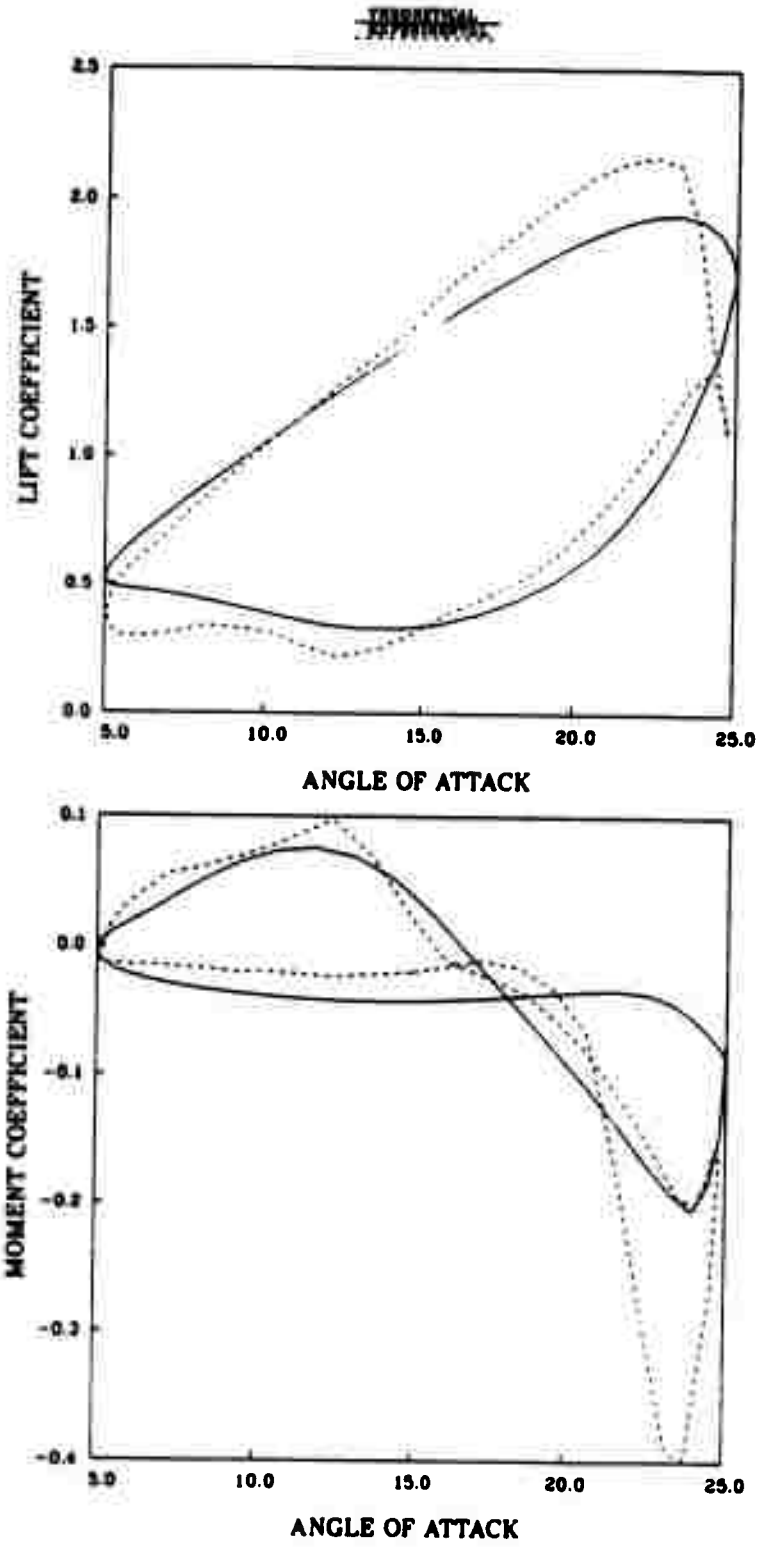


Figure 4.15 Lift and Pitching Moment Coefficients--Modified Model until 25°
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15° - 10° \cos(.3t)$.

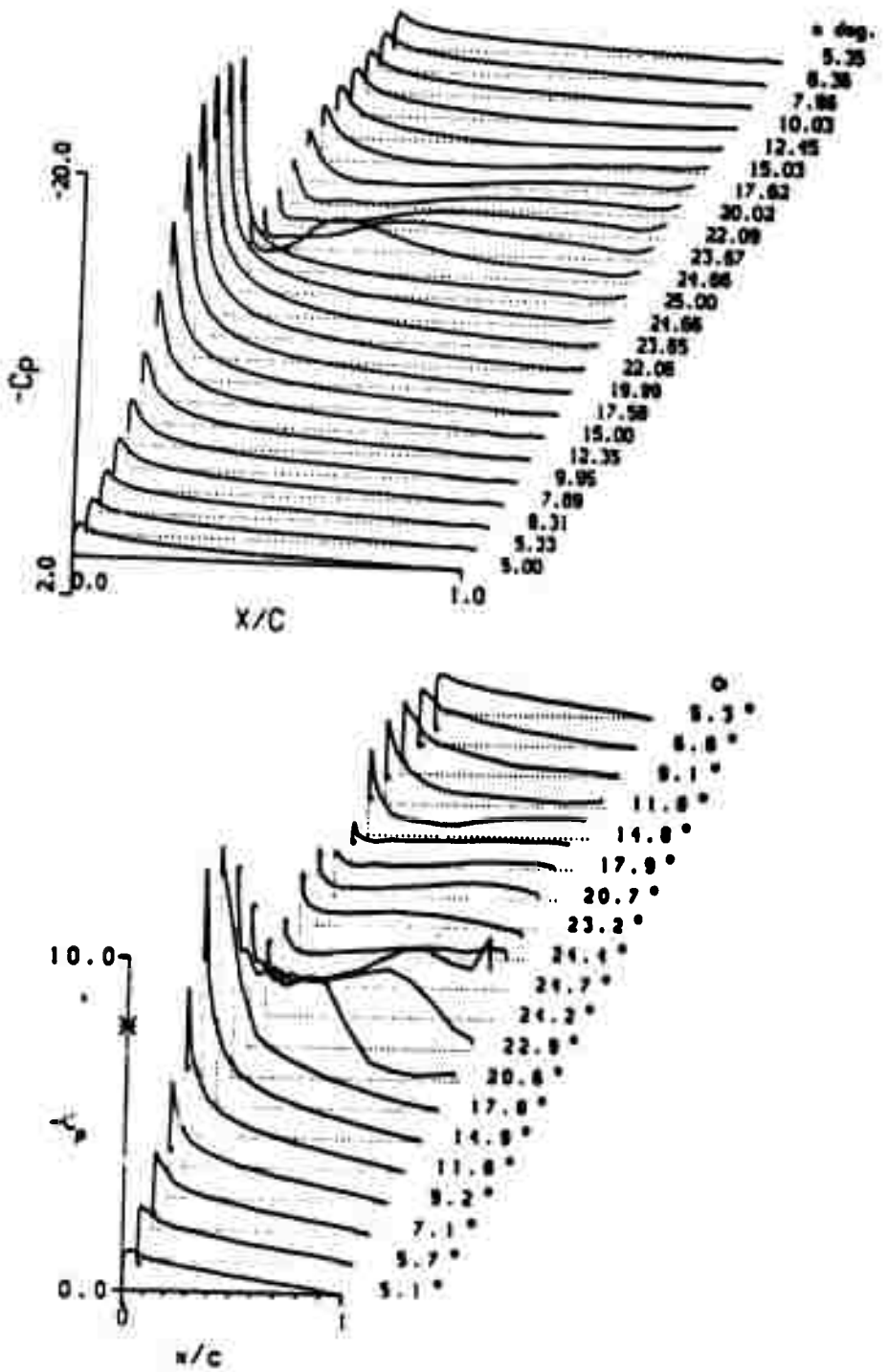


Figure 4.16 Surface Pressure Coefficient Carpet Plots--Modified Model until 25°
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15° - 10° \cos(.3t)$,
Top--theoretical, Bottom--experimental.

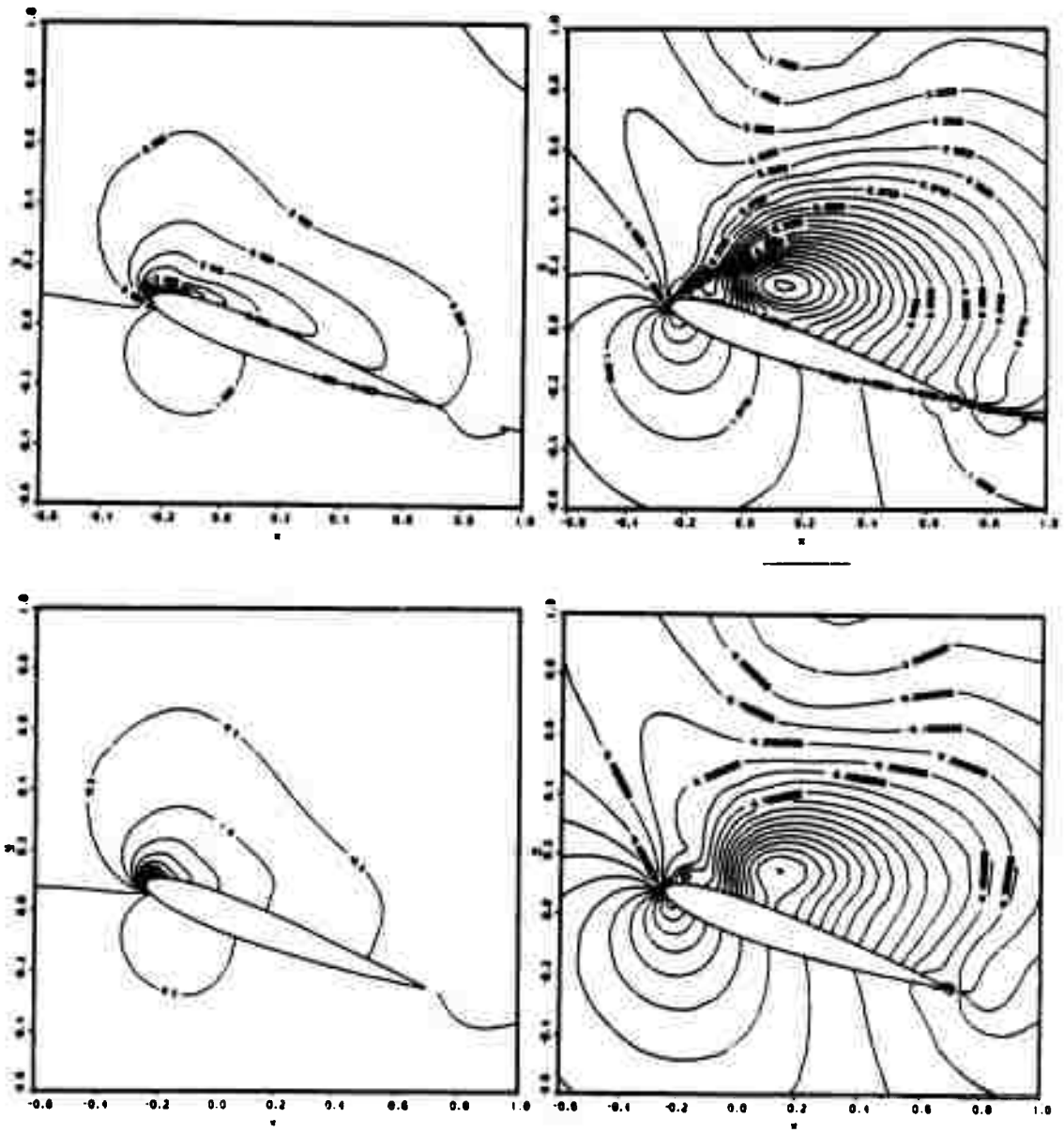


Figure 4.17 Density (top) and Pressure (bottom) Plots--Modified Model until 25°
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15° - 10° \cos(.3t)$
25° (l), 23.67° (r).

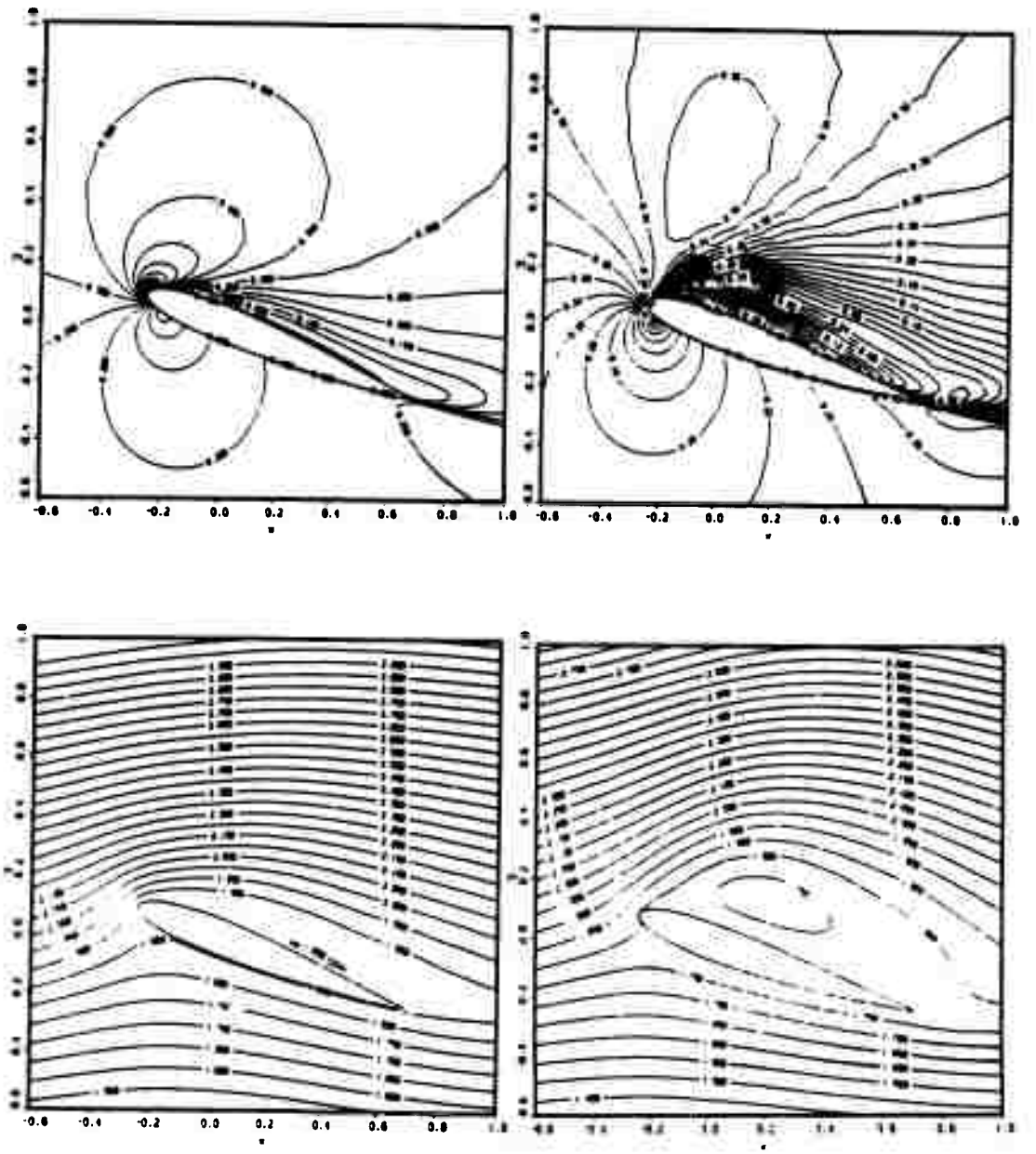


Figure 4.18 Mach Number and Stream Function Plots--Modified Model until 25°
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15° - 10° \cos(.3t)$
25° (l), 23.67° (r).

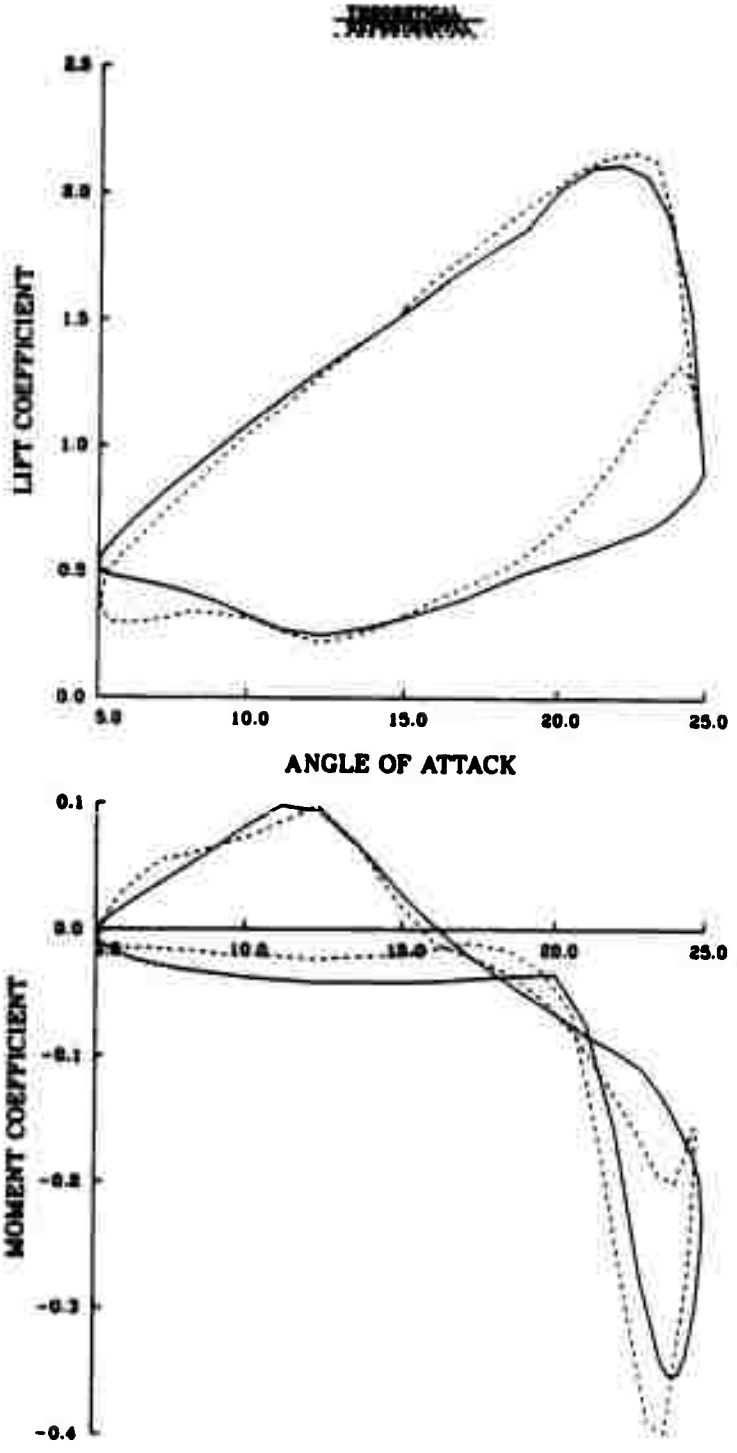


Figure 4.19 Lift and Pitching Moment Coefficients--Modified Model until 19°
$M_\infty = .283$, $Re = 3.45 \times 10^6$, $\alpha = 15° - 10° \cos(.3t)$.

sharper. With the time step set at .005 and higher damping (WW = 10), results were not as good guantitatively. Peak values were reached at near the same points, but lift reached only 1.94, and peak moment was badly underpredicted at −2.5. For a final comparison, the solution was restarted again for a short run from 15 degrees, but this time with the modified turbulence model turned off immediately. In all cases, a second leading-edge suction peak,indicative of vortex formation, was evident in the first output after the model was switched off. It seems from these results that the use of a smaller time step during the high angle portion of the upstroke is worth the added computational expense.

C. COMPARISON WITH ATTACHED FLOW EXPERIMENTAL DATA

The program was next run under the experimental conditions of Frame 9118 of Reference 8. These conditions were chosen because, in contrast to the previous experimental results, flow remained attached throughout the oscillation cycle. Sufficiently lowering the reduced frequency (as in Frame 9110 of Reference 8) would result in separated flow. The program was run under the following conditions:

$Re = 2.45 \times 10^6$

$M_\infty = .184$

$\alpha_0 = 8^\circ$

$\alpha_1 = 10^\circ$

$k = .199$

Again both turbulence models were tried, but for these conditions it was possible to complete both runs with a time step of .005 and explicit viscosity coefficient of 5. Output was again obtained after one and one-fourth cycles.

As was expected, the Baldwin-Lomax model gave a separated flow prediction during the upstroke. The first negative surface friction values appear near the trailing edge at 16.3 degrees, and near 17 degrees a vortex forms at the leading edge. The lift and moment plots of Figure 4.20 show this discrepancy from experimental results. The lift falls below experimental values at high angles, but again, the action of the vortex sustains the lift coefficient and even causes the slope of the lift curve to increase slightly before the downstroke begins. The downstroke shows similar behavior to the dynamic stall case investigated previously, with a sharp drop in lift, which remains at a low value until the upstroke begins. The moment coefficient is again low during the upstroke and oscillates as the vortex propagates along the airfoil surface. The carpet

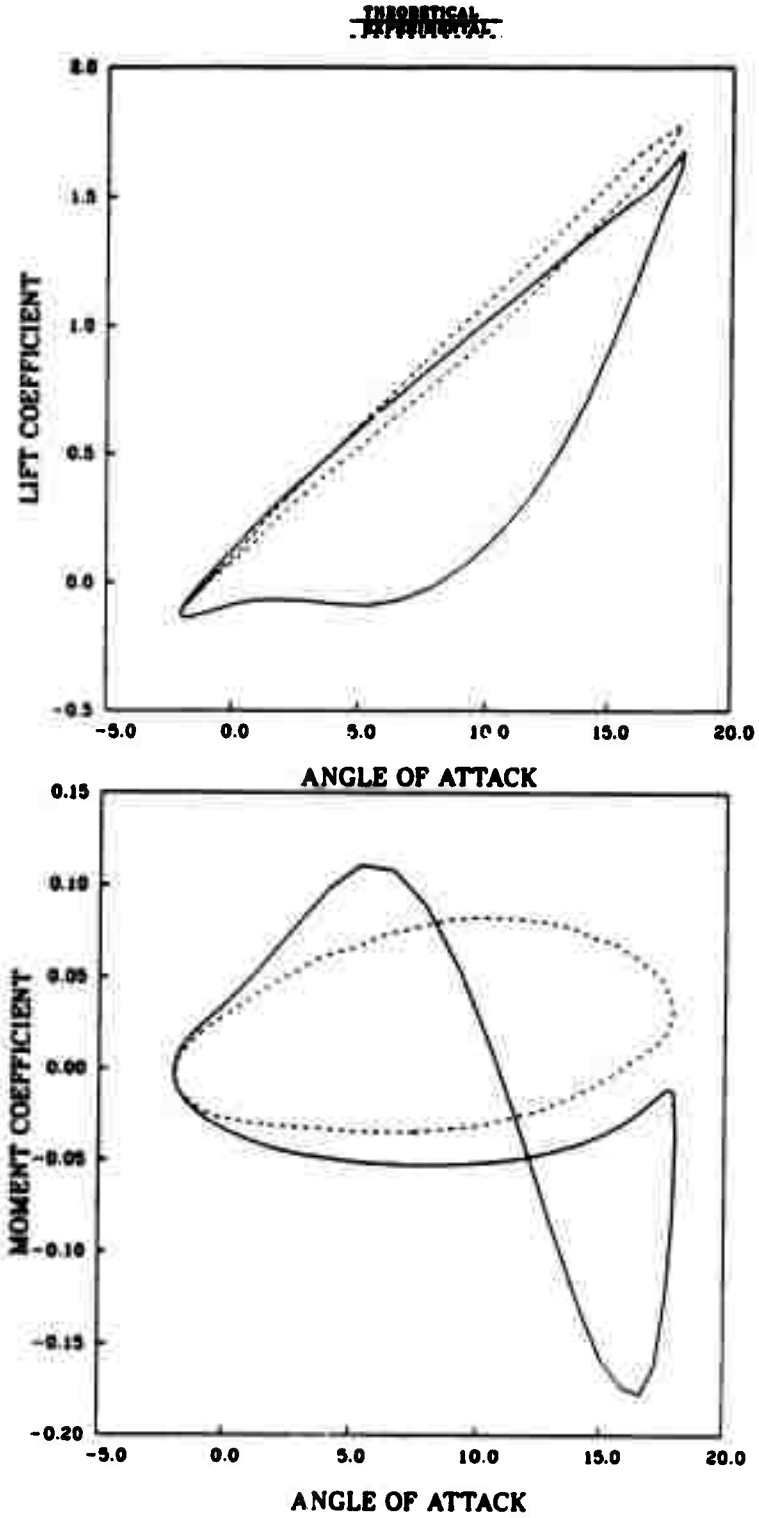


Figure 4.20 Lift and Pitching Moment Coefficients--Baldwin-Lomax Model

$M_\infty = .184$, $Re = 2.45 \times 10^6$, $\alpha = 8° - 10° \cos(.4t)$.

plot, Figure 4.21, shows the familiar ripple pattern of vortex movement. The flow field at the top of the cycle (18 degrees) is illustrated in Figure 4.22.

Based on the previous results with the modified turbulence model, it was expected to perform better under this set of conditions than the Baldwin-Lomax model. The flow does in fact remain attached during the upstroke but it separates as soon as the downstroke begins, as it did in the dynamic stall case when the modified model was used throughout the upstroke. The lift and drag curves of Figure 4.23 are very similar in appearance to those of Figure 4.20 except for the loops formed at the maximum angle by the rapid vortex formation on the downstroke. The carpet plot, Figure 4.24, agrees quite well with experiment until the maximum angle. The flow field is shown by Figure 4.25 to be very similar at 16.64 degrees on the downstroke to that for the Baldwin-Lomax model at the maximum angle (Fig. 4.22).

D. SIMULATIONS UNDER PROPOSED EXPERIMENTAL CONDITIONS

A series of dynamic stall wind tunnel experiments is presently being planned at the Ames Research Center Fluid Mechanics Laboratory. These experiments, which will include investigation of compressibility effects, are to be conducted in a Reynolds number range lower by a factor of ten than those reported in Reference 8. The last two simulations attempted, therefore, were made at a Reynolds number of 345,000. In order to have some basis for comparison, the mean angle, amplitude of oscillation, and reduced frequency chosen were the same as for the deep dynamic stall, high Reynolds number runs. The Mach numbers used were .284 and .5. The conditions then were the following:

$Re = 345{,}000$

$M_\infty = .284$ (Run 1), .5 (Run 2)

$\alpha_0 = 15°$

$\alpha_1 = 10°$

$k = .151$

The conditions are within the range of the planned experiments. The Baldwin-Lomax model was used for both runs. The first run was conducted with a constant time step of .005 and viscosity coefficient input of 5. At the higher Mach number of .5, the time step was set to .0025. An attempt to start the run with a .005 time step resulted in abnormal termination after fewer than 30 steps due to the calculation of negative

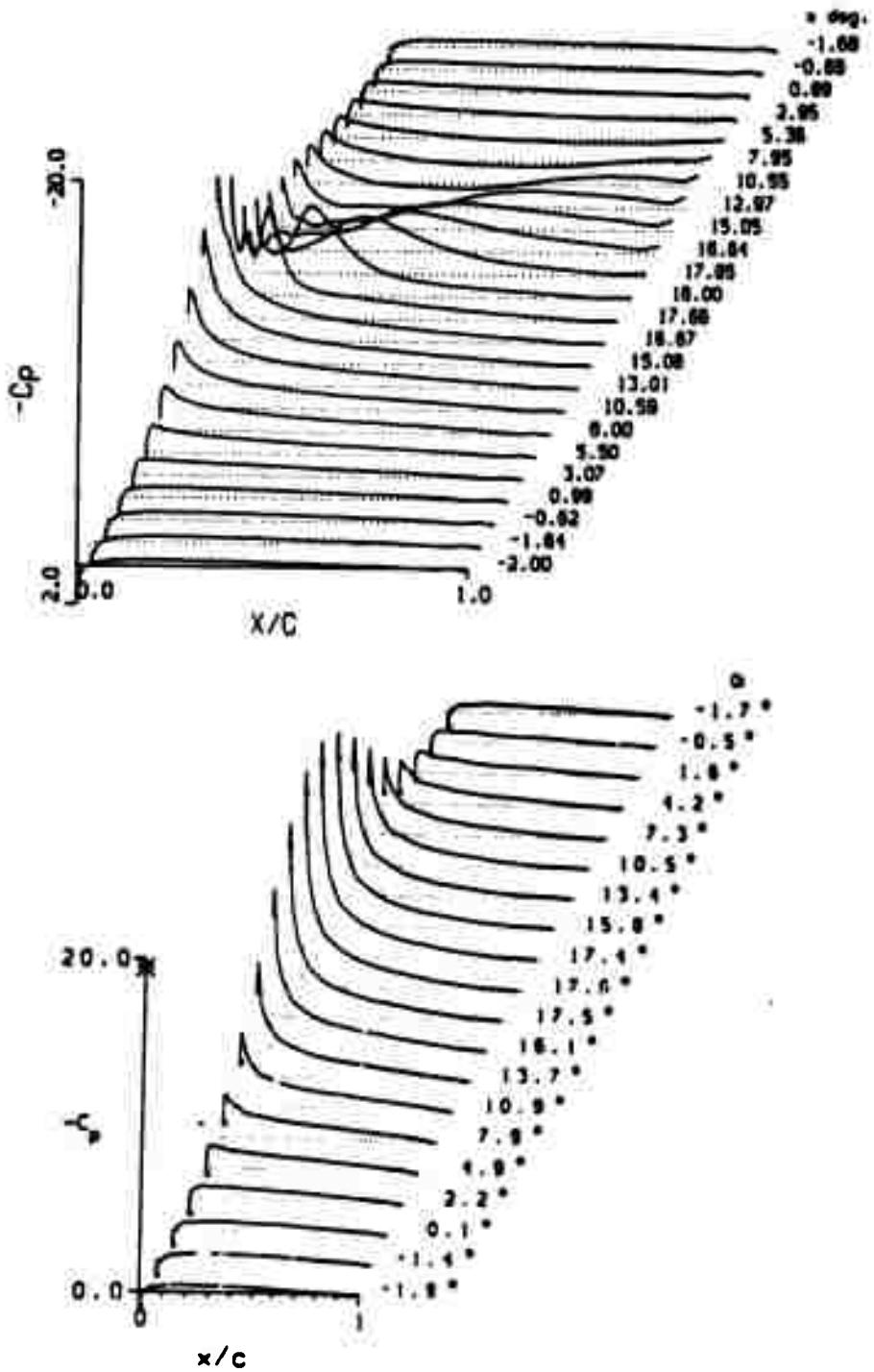


Figure 4.21 Surface Pressure Coefficient Carpet Plot--Baldwin-Lomax Model
$M_\infty = .184$, $Re = 2.45 \times 10^6$, $\alpha = 8^\circ - 10^\circ \cos(.4t)$
Top--theoretical, Bottom--experimental (Ref. 8).

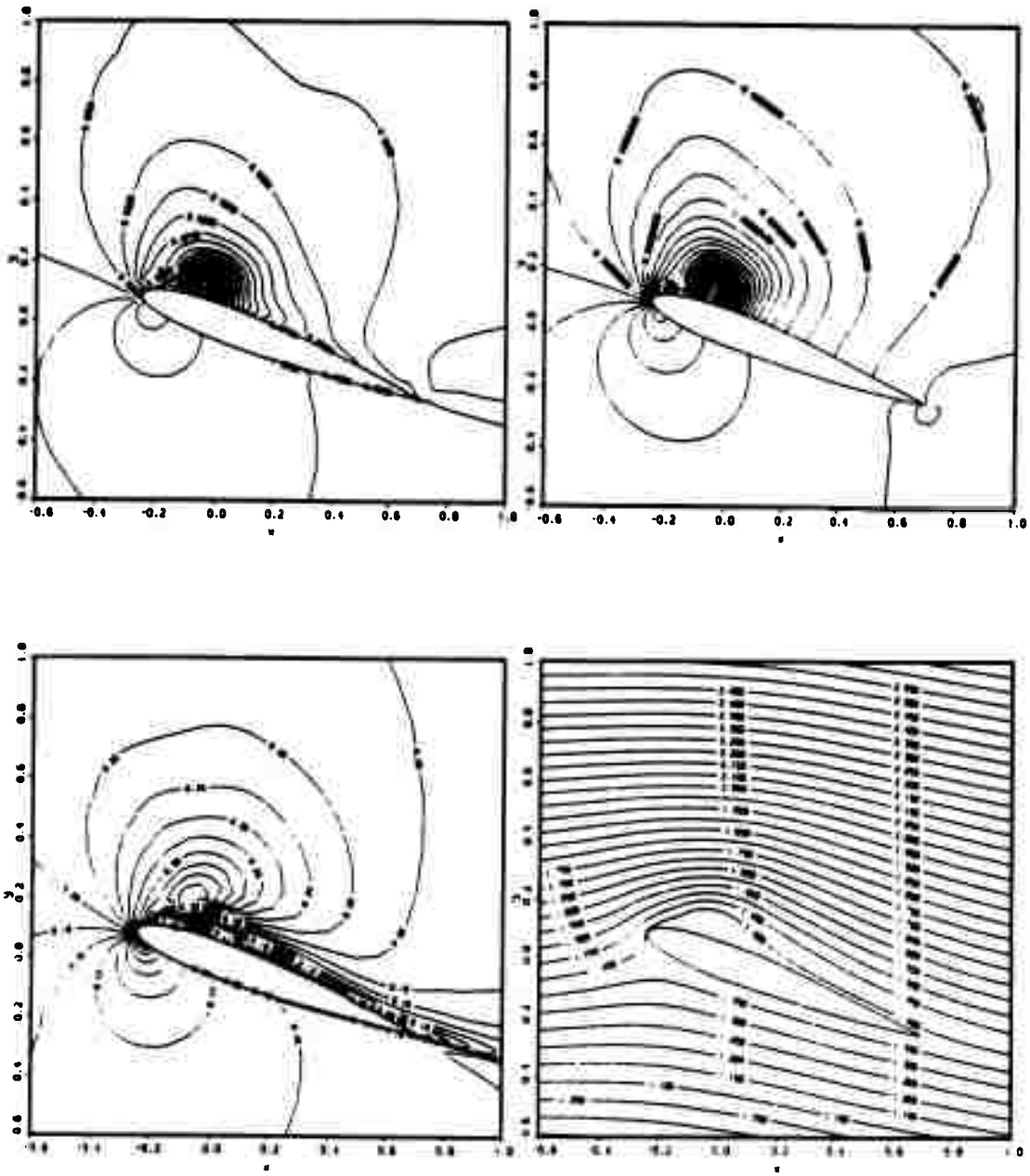


Figure 4.22 Flow-field Plots--Baldwin-Lomax Model
$M_\infty = .184$, $Re = 2.45 \times 10^6$, $\alpha = 8° - 10° \cos(.4t)$
Top--Density (l), Pressure (r) Bottom--Mach No. (l), Stream Function (r).

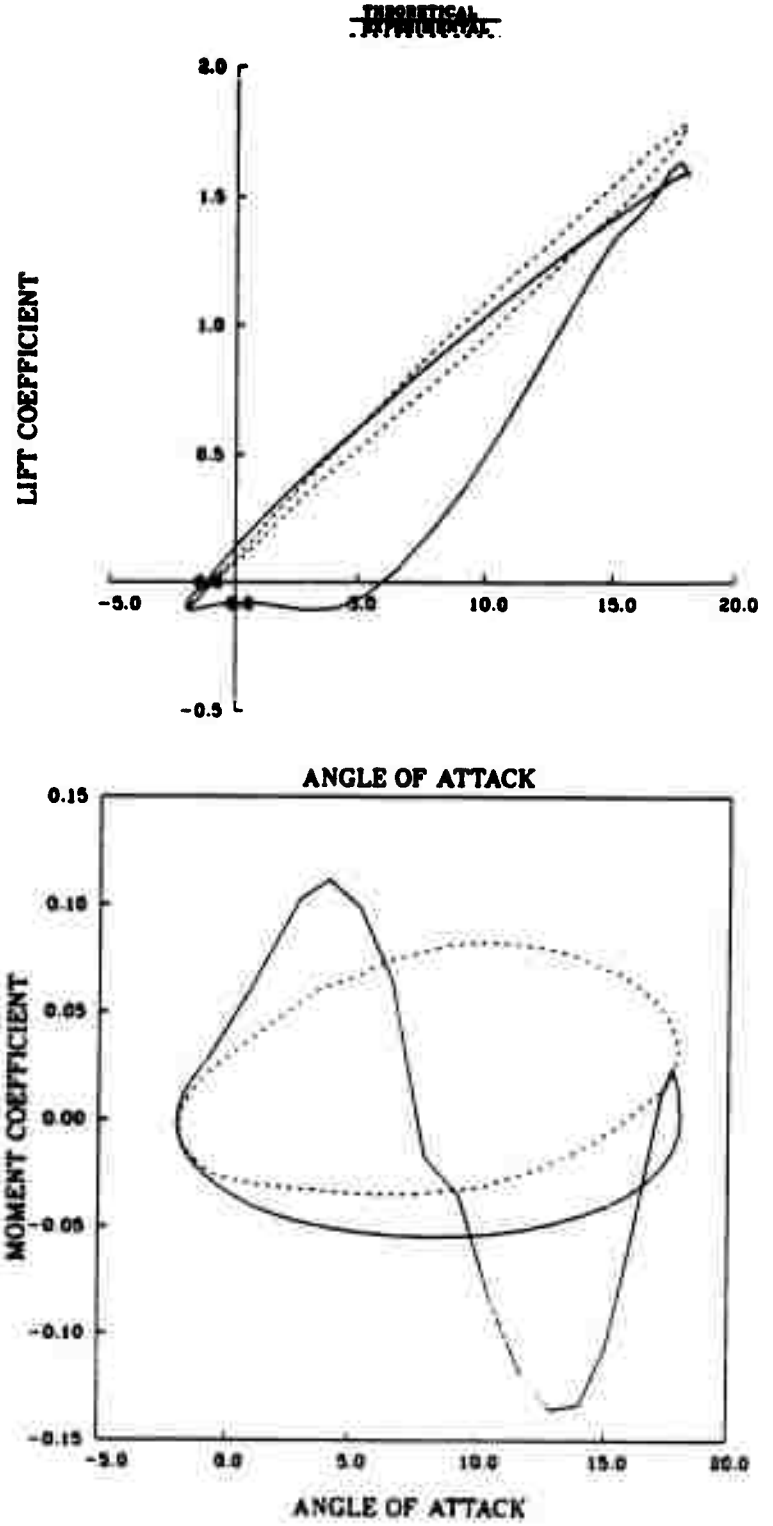


Figure 4.23 Lift and Pitching Moment Coefficients--Modified Turbulence Model $M_\infty = .184$, $Re = 2.45 \times 10^6$, $\alpha = 8° - 10° \cos(.4t)$.

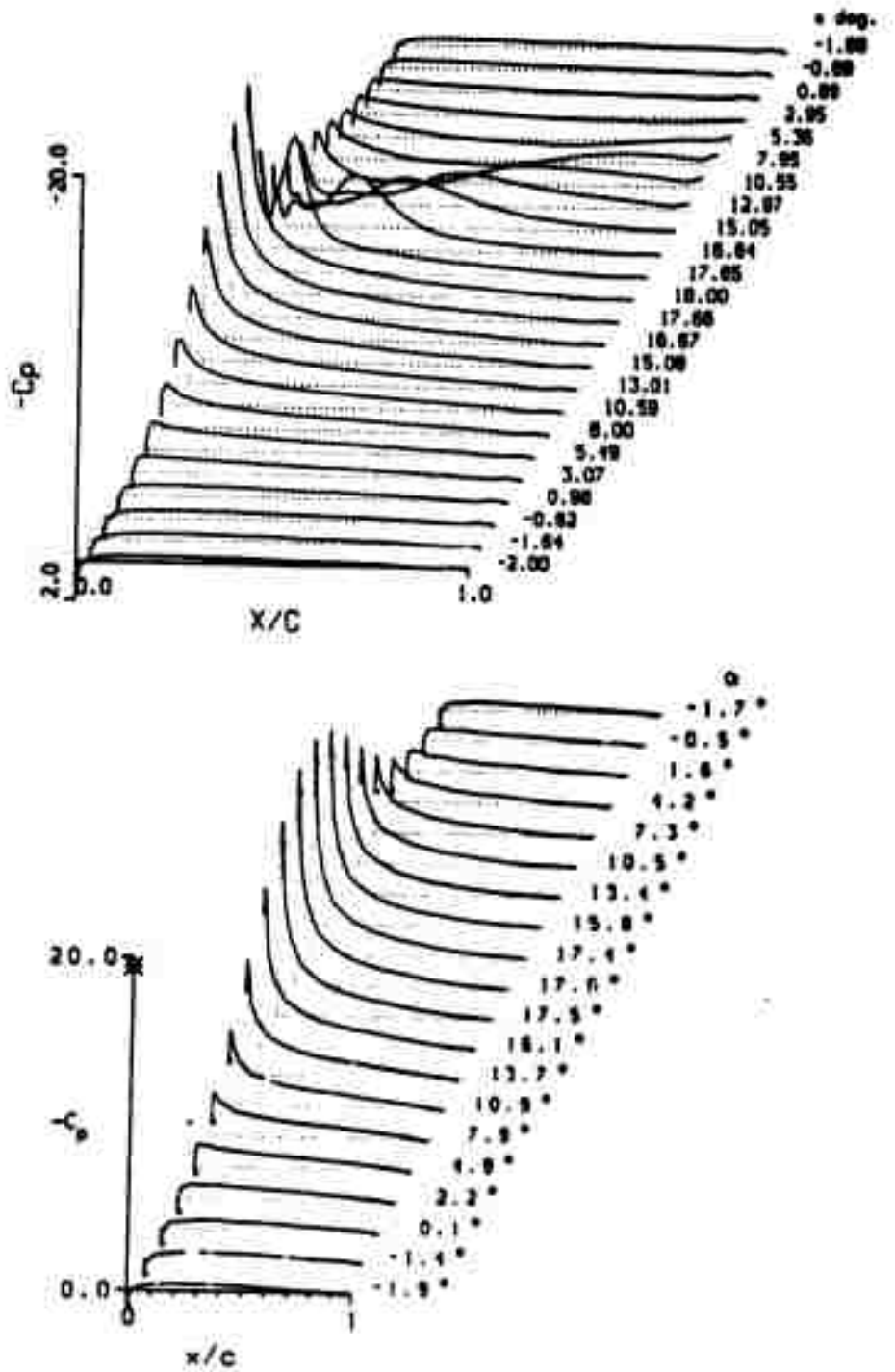


Figure 4.24 Surface Pressure Coefficient Carpet Plots--Modified Turbulence Model
$M_\infty = .184$, $Re = 2.45 \times 10^6$, $\alpha = 8° - 10° \cos(.4t)$
Top--theoretical,Bottom--experimental (Ref. 8).

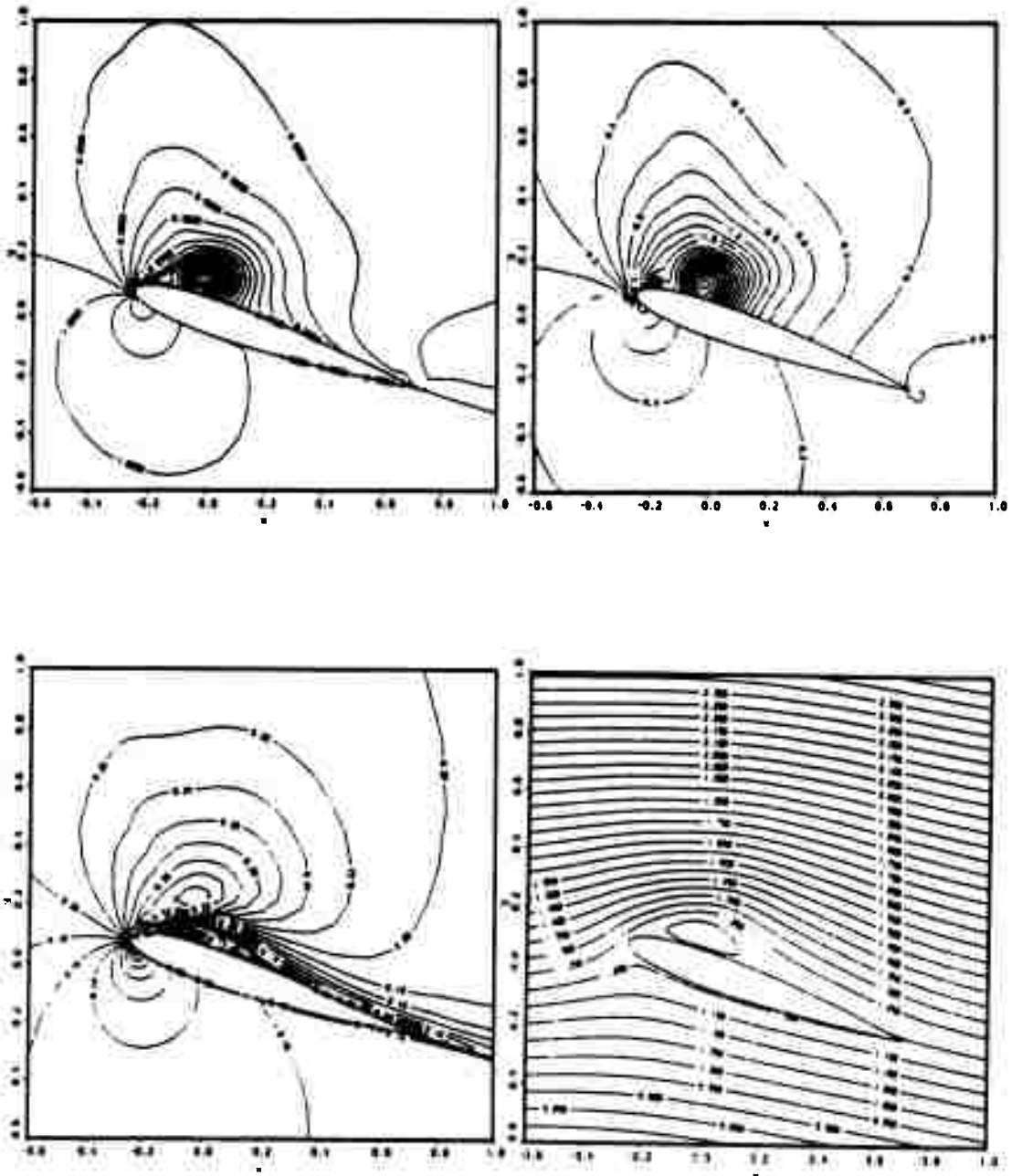


Figure 4.25 Flow-field Plots--Modified Turbulence Model
$M_\infty = .184$, $Re = 2.45 \times 10^6$, $\alpha = 8^\circ - 10^\circ \cos(.4t)$
Top--Density (l), Pressure (r) Bottom--Mach No. (l), Stream Function (r).

density near the leading edge, just as had happened during the earlier high Reynolds number run with the modified turbulence model. With the smaller time step, little improvement was realized until the artificial viscosity input was increased. Fewer than fifty steps were completed with the explicit viscosity at 5. When this was doubled the solution reached nearly 8 degrees angle of attack and completed 2170 steps before it again "blew up". In each instance in which this abnormal termination took place, the output values all appeared normal until the values of the residuals began growing shortly before termination. Increasing the explicit viscosity coefficient to 20 permitted a complete solution to be obtained.

The results for these two runs were qualitatively similar to those for the higher Reynolds number. The lift and moment coefficients plotted in Figure 4.26 show values close to those of the earlier results. The slopes of the lift curves are somewhat steeper than the previous results, and the lift continues to drop (and the moment coefficient to rise) farther into the downstroke. Figure 4.27 compares the pressure distribution at the lower Mach number with earlier results for the high Mach number. The suction peaks at the low Reynolds number are not as strong and begin breaking down slightly earlier. The flow-field plots showed a similar series of events to those seen previously and are not presented here.

The results obtained at the higher Mach number have two notable features: smoother contours and lower peak values. At least some of this smoothing must be attributed to the much higher viscosity input used for this case (20 versus 5). The lift and moment plots are given in Figure 4.28 and the carpet plot for this case along with that at the lower Mach number are shown for comparison in Figure 4.29. The shape of Figure 4.28 curves is more rounded than was Figure 4.26, and maximum lift is reduced. The pressure coefficient suction peaks (Fig. 4.29) are also smoother, lower, and begin breaking down sooner at the higher Mach number. To further investigate the effect of artificial viscosity, a partial run was made with the explicit damping coefficient input as 55. The maximum lift coefficient obtained was 1.3 at 17.6 degrees angle of attack compared to 1.6 at 18.2 degrees with the viscosity at 20. The maximum value was attained at nearly the same angle (output was not taken at exactly the same points for this second run). However, the first irregularities in the leading-edge pressure coefficient were delayed by this increase in viscosity from about 12.3 degrees angle of attack to near 15 degrees. Thus, the computer results do seem to be predicting an earlier vortex formation (and breakdown in the surface pressure coefficient) at the higher Mach number.

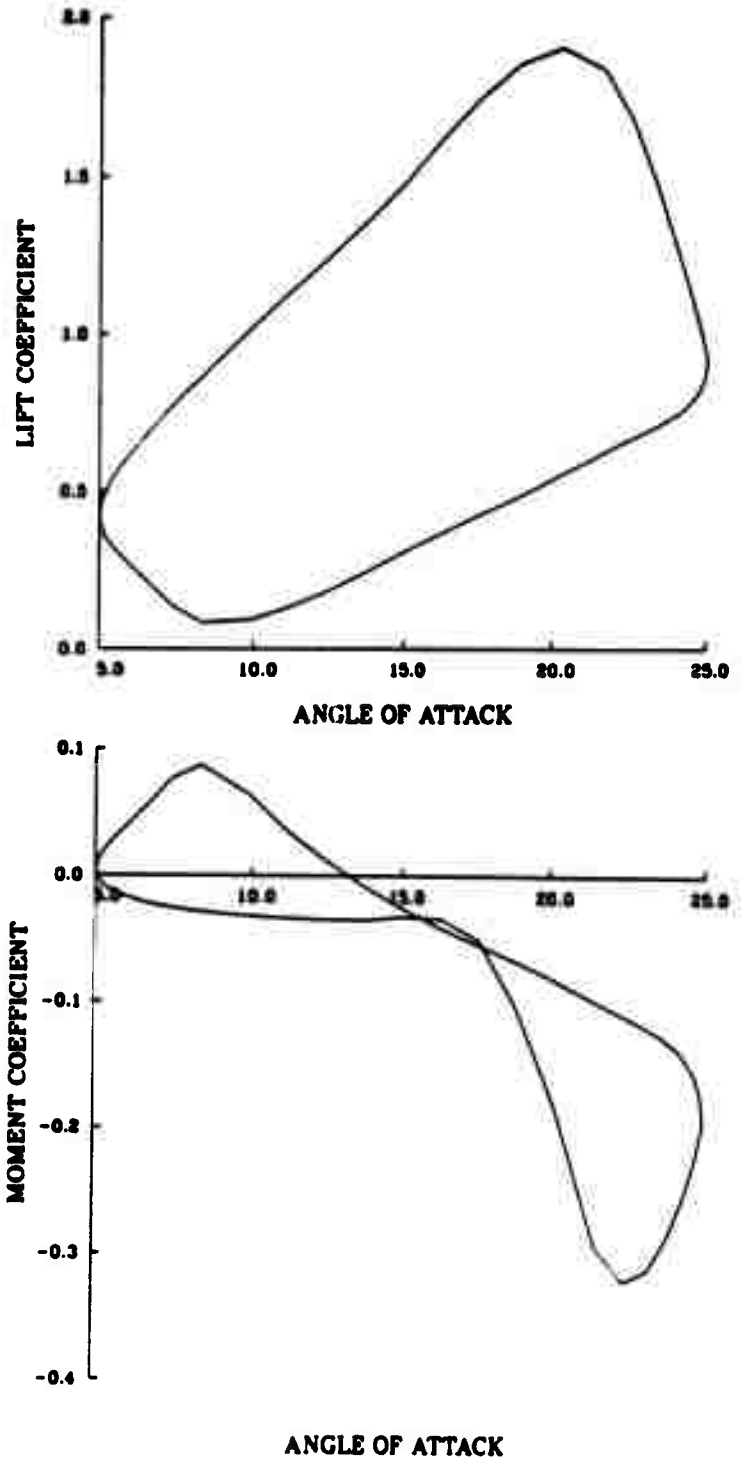


Figure 4.26 Lift and Pitching Moment Coefficients--Baldwin-Lomax Model $M_\infty = .284$, $Re = .345 \times 10^6$, $\alpha = 15^\circ - 10^\circ \cos(.3t)$.

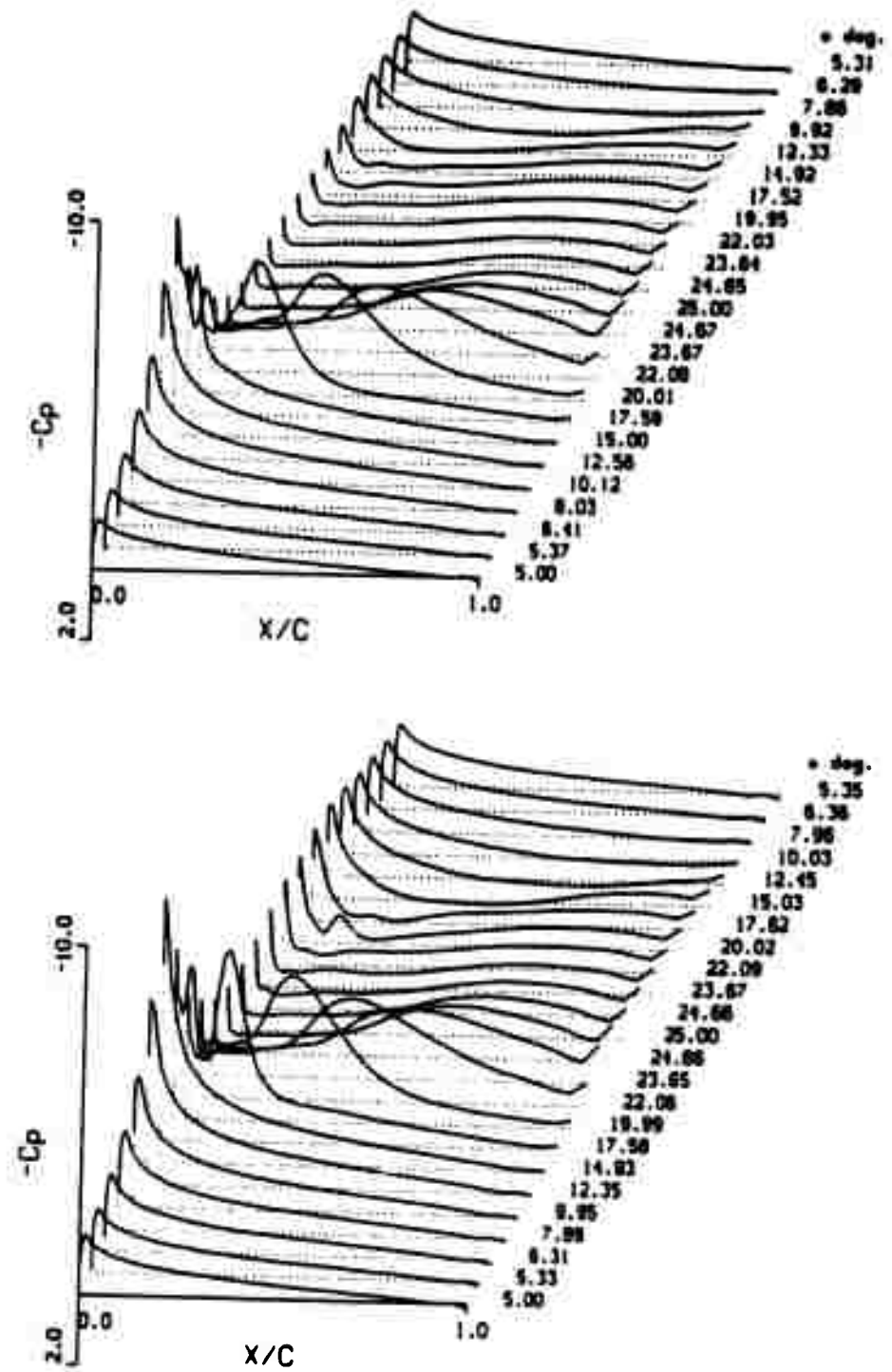


Figure 4.27 Surface Pressure Coefficient Carpet Plots--Baldwin-Lomax Model
Top: $M_\infty = .284$, $Re = .345 \times 10^6$, $\alpha = 15^\circ - 10^\circ \cos(.3t)$
Bottom: $Re = 3.45 \times 10^6$(Previous Results).

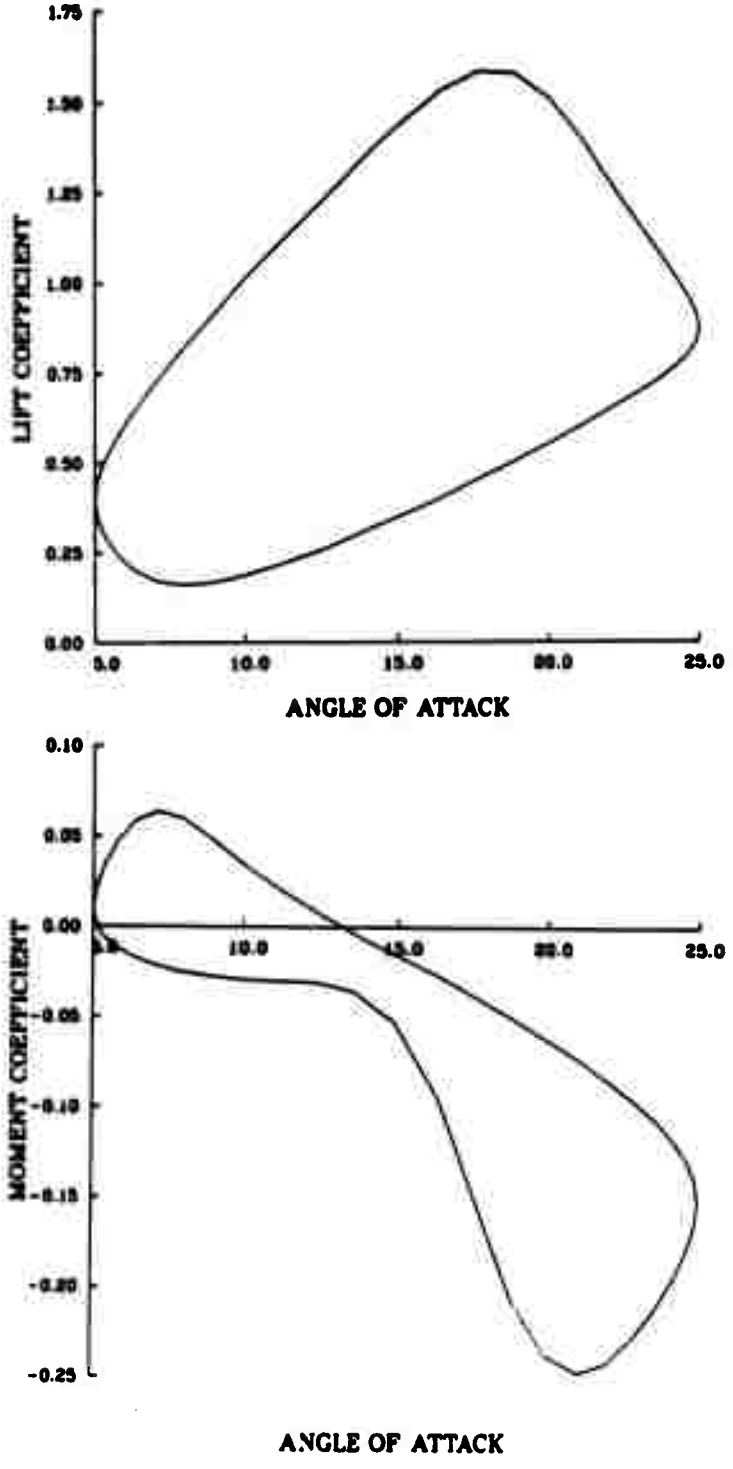


Figure 4.28 Lift and Pitching Moment Coefficients--Baldwin-Lomax Model $M_\infty = .5$, $Re = .345 \times 10^6$, $\alpha = 15^\bullet - 10^\bullet \cos(.3t)$.

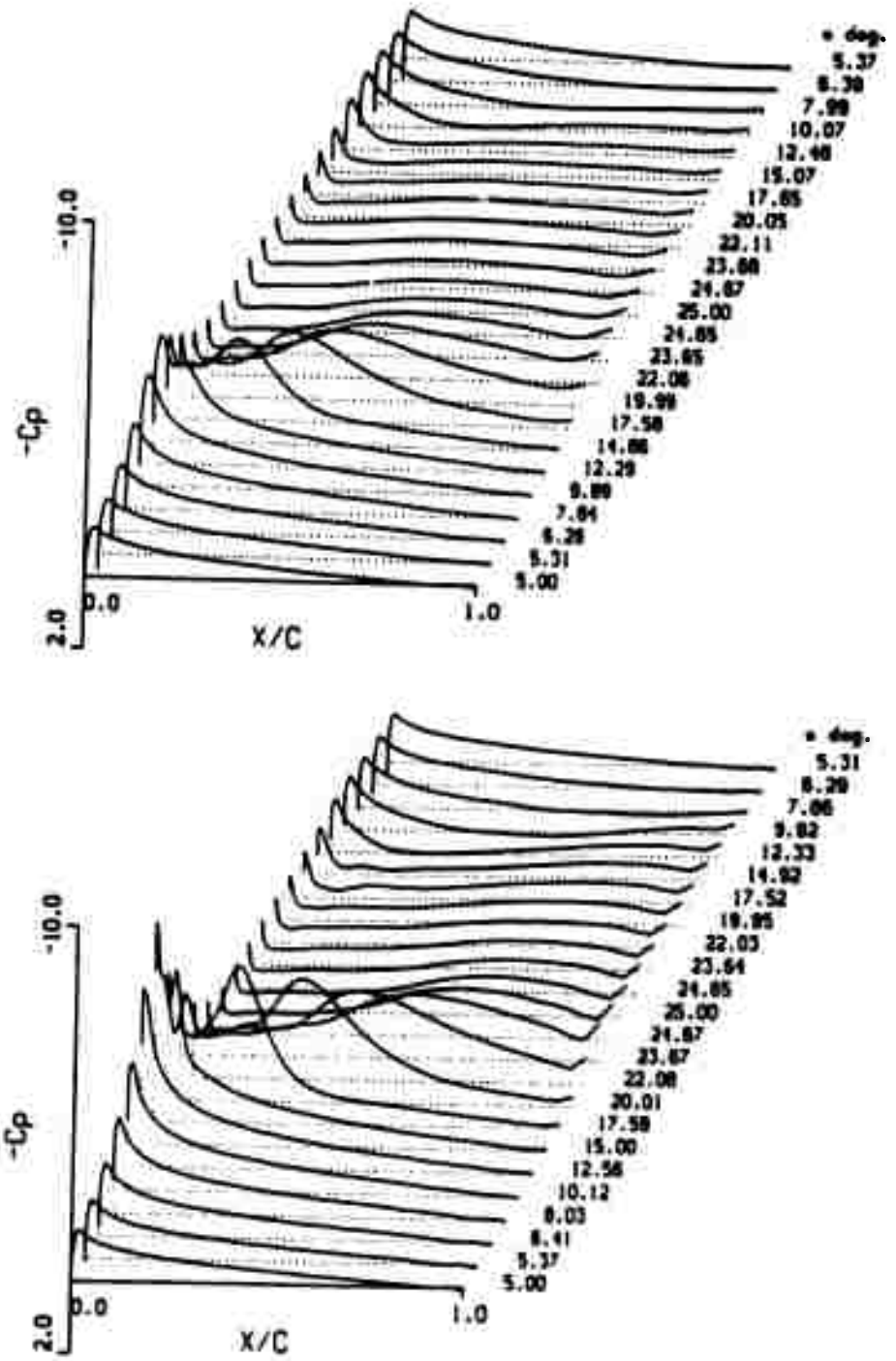


Figure 4.29 Surface Pressure Coefficient Carpet Plots--Baldwin-Lomax Model
Top: $M_\infty = .5$, $Re = .345 \times 10^6$, $\alpha = 15^\circ - 10^\circ \cos(.3t)$
Bottom: $M_\infty = .284$ (Previous Results).

V. CONCLUDING REMARKS

The computer program employed for this study has shown the capability to model the basic events of the dynamic stall process. The quality of the results is, not surprisingly, highest at moderate angles--two to three degrees above static stall angle in examples studied. Results at higher angles are strongly dependent on turbulent viscosity model. The downstroke is not as well approximated, although there is qualitative agreement of the integrated coefficients. The program appears quite robust at Mach numbers of .3 and below using the Baldwin-Lomax turbulence model. With the modification to the turbulence model introduced in this implementation, better results are obtainable at the higher angles, but the program is less stable. Adequate convergence was generally achieved quickly enough that data could be saved beginning at the mean angle on the first oscillation cycle.

The examples considered show some of the limitations of current Navier-Stokes solvers. By manipulating artificial viscosity and turbulence modelling, it is possible to manage calculations to match, in some degree, desired results. For example, the attached flow case considered might be brought into closer agreement with experiment by using the modified turbulence model after the beginning of the downstroke. For the dynamic stall case, however, this might lead to premature flow reattachment and less accurate results. The simple modification introduced, used as recommended during the upstroke, shows encouraging results. At no significant expense, modelling near and a few degrees above static stall angles appears improved without affecting results at lower angles. Further testing needs to be done to determine the effectiveness of the change under different dynamic conditions, and the use of other models is certainly worth consideration. King and Johnson have reported favorable results with a nonequilibrium turbulence model using an ordinary differential equation to model streamwise shear stress development, and an eddy viscosity distribution across the boundary layer based on the maximum value from this equation. Compared to the Baldwin-Lomax and Cebeci-Smith models, it showed little difference in attached flow calculations with a Navier-Stokes solver using a Beam-Warming scheme. When applied to separated flows and flows with strong viscous effects, it gave superior results, though at some computational expense in its present form. [Refs. 18,20]

The use of other grids or variations on the present one would be worth considering for the purpose of developing comparisons for the proposed experiments at the Ames Research Center Fluid Mechanics Laboratory. Obtaining a solution at the .5 Mach number was difficult and required a large increase in the artificial viscosity. Some of the problems might be alleviated by closer spacing in the leading-edge area where the instability arose. The present grid generation scheme might be tried with more points selected in the normal direction. Events at the trailing edge are not now modelled with complete accuracy. The vortex moves to the trailing edge and diffuses there. This behavior may be influenced by the level of artificial viscosity (which must be increased to compensate for grid coarseness), the downstream boundary, location of the grid cut, and relative coarseness of the grid at the trailing edge. The present grid provides adequate results for a large range of applications with a reasonable number of grid points, however.

The solver now assumes a fully turbulent boundary layer (or fully laminar if the Reynolds number is set to zero). The proposed experiments may include significant transition point effects. The transition could be simulated by retaining the laminar viscosity coefficient forward of a specified chord location. This might be especially useful for comparison with tripped boundary layer data. Some runs were made with fully laminar viscosity. The pressure gradients at the leading edge could not be sustained, and multiple vortices were formed but the solution remained stable.

The cases considered in this study represent useful test conditions for studying the effect of changing various program segments. The deep stall case has the model dynamic stall features and has been used by Sankar. The attached flow case is an interesting and a difficult contrasting case. The forthcoming experiments will provide the opportunity to investigate details of the dynamic stall process, including compressibility effects, in much greater depth. If the Navier-Stokes solver is not yet a completely adequate predictive tool, the effort to make it so can, of itself, provide insight into the nature of unsteady flows.

APPENDIX A

SANKAR NAVIER-STOKES SOLVER

```
JOB,JN=,T=.
ACCOUNT,AC=,US=,UPW=.
*.
CFT,ON=A,OFF=S.
*.
*.ASSIGN,DN=NEWSLN,A=FT08.
*.ASSIGN,DN=NEWCP,A=FT18.
*.ASSIGN,DN=XYZ,A=FT11.
*.ASSIGN,DN=Q,A=FT12.
*.IS THIS RUN TO START FROM A STORED SOLUTION?
*.ACCESS,DN=OLDSLN,PDN=,ID=.
*.DO YOU WANT TO ACCESS EXISTING PRESSURE COEFFICIENT DATA?
*.ACCESS,DN=OLDCP,PDN=,ID=.
*.
*.ASSIGN,DN=OLDSLN,A=FT07.
*.ASSIGN,DN=OLDCP,A=FT17.
*.
LDR,MAP=PART,SET=ZERO.
*.DO YOU WANT TO SAVE THE CURRENT SOLUTION?
*.SAVE,DN=NEWSLN,PDN=,ID=.
*.DO YOU WANT TO SAVE PRESSURE COEFFICIENT DATA?
*.(PREVIOUS FILES SHOULD BE PURGED FREQUENTLY).
*.SAVE,DN=NEWCP,PDN=,ID=.
*.
*.DO YOU WANT TO CREATE PLOT3D FILES?
*.ACCESS,DN=SENDVAX,PDN=SENDVAX,ID=STTRDM.
*.SENDVAX,DN=XYZ,VDN=.
*.SENDVAX,DN=Q,VDN=.
*.
*.DO YOU WANT PLOT3D FILES FOR TROUBLESHOOTING IF PROGRAM CRASHES?
*.EXIT.
*.ACCESS,DN=SENDVAX,PDN=SENDVAX,ID=STTRDM.
*.SENDVAX,DN=XYZ,VDN=.
*.SENDVAX,DN=Q,VDN=.
*.
*.DO YOU WANT TO USE JOB CHAINING?
*.FETCH,DN=,TEXT=.
*.REWIND,DN=.
*.SUBMIT,DN=.
/EOF
      PROGRAM MAIN
*     PROGRAM MAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      COMMON/SURF/PSUR(161)
      COMMON/FIX/OMEGA
      COMMON/MUTUR/CMU(161,41)
      COMMON/SKINCF/CF(161)
      COMMON/GRID1/X(161,41),Z(161,41)
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
```

```
      COMMON/GRID/YACOB(161,41)
      COMMON/DAMP/WW,WWI
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      DIMENSION TITLE(15),TYTLE(15),ALPHA(96),CPTH(97,96),XTH(97)
      COMMON/LOGIC/RSTRT,PITCH,PLUNGE
      LOGICAL RSTRT,PITCH,PLUNGE
C
C!!!!!!INDICATES COMMENTS OR CHANGES ADDED BY J.F. VALDES, DEC 1986
C***  PROGRAM SOLVES TWO-DIMENSIONAL FLOW PAST ARBITRARY
C***  GEOMETRIES USING ADI PROCEDURE.
C
C     TAPE5 = FILE CONTAINING INPUT DATA
C     TAPE6 = OUTPUT
C     TAPE8 = FILE THAT SAVES THE FLOW FIELD AT THE END OF A RUN
C     IF THE CURRENT RUN IS A RESTART OF A PREVIOUS RUN, THEN
C     TAPE7 IS USED TO READ THE FLOW FIELD INTO MEMORY
C!!!!!!TAPE18 IS USED TO ACCUMULATE PRESSURE COEFFICIENT DATA FOR A CYCLE
C!!!!!!   IT MUST BE ACCESSED WHEN COMPLETE BY PROGRAM PLOTNSE.SRC
C!!!!!!TAPE17 IS USED TO READ EXISTING PRESSURE COEFFICIENT DATA FOR
C!!!!!!   UPDATE DURING THE CURRENT PROGRAM RUN
C!!!!!!TAPE11 IS USED FOR PLOT3D XYZ-FILE OUTPUT
C!!!!!!TAPE12 IS USED FOR PLOT3D Q-FILE OUTPUT
C
C***  READ INPUT DATA
C
      PI = ATAN(1.)*4.
      READ (5,1) TITLE
      READ (5,2) IMAX,KMAX,DT,WW,ALFA,ALFA1,ALFAI,REDFRE,AMINF
C!!!!!SET ALFAD FOR STEADY STATE PLOT3D OUTPUT
      ALFAD = ALFA
C
C      FORCE DT TO BE EQUAL TO UNITY FOR STEADY FLOW PROBLEMS
C      THIS INVOKES THE RELAXATION MODE
C
      IF(REDFRE.LE.0.001) DT = 1.0
      READ (5,2001)
C      NSTP = NO. OF TIME STEPS TO BE DONE ON THIS RUN
      READ (5,2221) FNSTP
      NSTP = FNSTP
      READ (5,2001)
C     REYREF= REYNOLDS NUMBER IN MILLIONS
C     DNMIN = DISTANCE OF FIRST POINT OFF THE WALL
C     FOR REYNOLDS NUMBERS UPTO 3 MILLION USE 0.00005
      READ (5,2221) REYREF,DNMIN
      REYREF = REYREF * 1.E+06
C
C    TSTART = TIME THAT THE CALCULATIONS HAVE BEEN ADVANCED
C    UPTO THE PREVIOUS RUN. IF TSTART IS NEGATIVE THIS VALUE IS
C    OBTAINED FROM TAPE 8.
      READ (5,2001)
      READ (5,2221) TSTART
C!!!!!FULOUT IS A FLAG FOR GENERATING PLOTTING FILES. WHEN NEGATIVE
C!!!!!NO DATA IS GENERATED. ZERO IS USED TO START AND A POSITIVE
C!!!!!NUMBER TO CONTINUE GENERATING FULL OUTPUT. ADDED FEATURE(JFV).
      READ (5,2001)
      READ (5,2221) FULOUT
 2221 FORMAT(2F10.4)
      READ (5,2001)
 2001 FORMAT(1X)
      READ (5,2000) RSTRT,PITCH,PLUNGE
```

```
 2000 FORMAT(4L5)
C      NEGATIVE REYREF MEANS INVISCID FLOW
C
C***  PRINT OUT THE INPUT DATA
C
      WRITE (6,4) TITLE
      WRITE (6,3) IMAX,KMAX,DT,WW,ALFA,ALFA1,ALFAI,REDFRE,AMINF
      IF(REYREF.GT.0.) WRITE (6,3700) REYREF
      GAMMA=1.4
C
C!!!!!GENERATE COMPUTATIONAL GRID
C
      CALL AIRFOL(IMAX,KMAX,ITEL,ITEU)
      IF(REYREF.GT.0.)CALL CLUSTR(DNMIN)
      WRITE (6,3002)
C
C***  STARTING CONDITIONS.
C***  DENSITY NORMALISED WITH RESPECT TO ROINF
C***  VELOCITIES NORMALISED WITH RESPECT TO AINF
C***  TOTAL ENERGY NORMALISED WITH RESPECT TO (ROINF*AINF*AINF)
C
      TOTEN=AMINF*AMINF*0.5+1./(GAMMA*(GAMMA-1.))
      ALFA = ALFA * ATAN(1.) / 45.
      ALMEAN = ALFA
      ALFAI = ALFAI * ATAN(1.) / 45.
      ALFA1 = ALFA1 * ATAN(1.) / 45.
      ALFAS = ALMEAN - ALFA1
      UINF = AMINF * COS(ALFA)
      VINF = AMINF * SIN(ALFA)
      DO 7 I=1,IMAX
      DO 7 K=1,KMAX
         Q1(I,K)=1.
         Q2(I,K)=UINF
         Q3(I,K)=VINF
         Q4(I,K)=TOTEN
    7 CONTINUE
      IF(RSTRT) READ (7) TIME,Q1,Q2,Q3,Q4
      IF(TSTART.GE.0.) TIME = TSTART
      IF(.NOT.(RSTRT)) TIME = 0.
C
C!!!!!READ STORED PRESSURE COEFFICIENTS. STATEMENTS ADDED BY JFV.
C
      IF (FULOUT .GT. 0.) THEN
         READ (17,1) TITLE
         READ (17) AMPLTD,STMEAN,OMS,XTH,ALPHA,CPTH
      END IF
C!!!!!BEGIN PRESSURE COEFFICIENT DATA FILE. STATEMENTS ADDED BY JFV.
      IF (FULOUT .EQ. 0.) THEN
         AMPLTD = ALFA1 * 45. / ATAN(1.)
         STMEAN = ALFA * 45. / ATAN(1.)
         OMS = REDFRE
         TEDGE = X(ITEL+48,1)
         DO 15 I=1,97
            XTH(I) = X(I+ITEL-1,1) - TEDGE
   15 CONTINUE
      END IF
C!!!!!DETERMINE NUMBER OF TIME STEPS OF CURRENT SIZE IN ONE CYCLE
C!!!!!  SCHEME FOR PLOTTING DATA AND PROGRAM EXIT ADDED BY JFV
      IF (PITCH) CYCLE = PI / (REDFRE * AMINF * DT)
C     ****
```

```
C
      CALL METRIC
      DO 1000 ITN=1,NSTP
C
C!!!!!CREATE PLOT3D FILES FOR INVESTIGATING INSTABILITY ADDED BY JFV
C!!!!!     THIS BLOCK UPDATES PLOT3D OUTPUT FILE AFTER
C!!!!!     EACH STEP. USE IN CONJUNCTION WITH EXIT JCL.
C!!!!! **** NOTE - USE ONLY FOR TROUBLESHOOTING ****
C
C         REWIND(11)
C         WRITE(11) IMAX, KMAX
C         WRITE(11) ((X(I,J), I=1,IMAX), J=1,KMAX),
C     1             ((Z(I,J), I=1,IMAX), J=1,KMAX)
C         REWIND(12)
C         WRITE(12) IMAX, KMAX
C         WRITE(12) AMINF, ALFAD, REYREF, TIME
C         WRITE(12) ((Q1(I,J), I=1,IMAX), J=1,KMAX),
C     1             ((Q2(I,J), I=1,IMAX), J=1,KMAX),
C     2             ((Q3(I,J), I=1,IMAX), J=1,KMAX),
C     3             ((Q4(I,J), I=1,IMAX), J=1,KMAX)
C
         TIME = TIME + DT
C!!!!!ROTATE GRID TO NEW ANGLE OR SET TO CORRECT ANGLE FOR RESTARTS
         IF (PITCH) THEN
            OMEGA = 2. * REDFRE *AMINF*SIN(REDFRE * 2. * TIME * AMINF)
     1      *ALFA1
            ALOLD = ALMEAN - ALFA1 * COS(2. * REDFRE * AMINF *
     1      (TIME - DT))
            ALFA = ALMEAN - ALFA1 * COS(REDFRE * 2. * TIME * AMINF)
            ALFAD = ALFA * 45. / ATAN(1.)
            DALFA = ALFA - ALOLD
            IF(RSTRT.AND.ITN.EQ.1) DALFA = ALFA - ( ALMEAN - ALFA1 )
            CALL ROTGRID(X,Z,IMAX,KMAX,DALFA)
            CALL METRIC
         END IF
         IF (PLUNGE) THEN
            OMEGA = 2. * REDFRE * AMINF
            ALFA = ALMEAN
         END IF
C
C!!!!!COMPUTE THE SOLUTION BY ADI TECHNIQUE
         CALL SLPS(ITN,OMEGA,DALFA)
C
C     APPLICATION OF BOUNDARY CONDITIONS
C
         CALL WALLBC
C
C     PRINT OUT PRESSURE AT THE SURFACE
C
C!!!!!FOR STEADY STATE OUTPUT USE THE FOLLOWING
         IF((ITN/100*100.EQ.ITN) .AND. (.NOT.PITCH)) THEN
            WRITE (6,19)
            WRITE (6,33) ITN
            CALL CPPLOT(ITEL,ITEU,AMINF,X(1,1),Z(1,1),PSUR)
            WRITE (6,12) (I,CF(I),I=1,IMAX)
            CALL LOAD(PSUR,CL,CD,CM,ALFAS)
            WRITE (6,3000) CL, CD, CM
            WRITE (6,3002)
         END IF
C!!!!!OR, FOR DYNAMIC SOLUTION OUTPUT AT EQUAL PHASE INTERVALS
```

```
C!!!!!   MODIFIED OUTPUT SCHEME AND PROGRAM EXIT SCHEME BY JFV.
         IF (PITCH) THEN
            TINCR = TIME/DT - CYCLE/4.
            IF (TINCR .LT. 0.) TINCR = TINCR + CYCLE
C!!!!!DETERMINE NUMBER OF STEPS BETWEEN OUTPUT
            NCPOUT = NINT(CYCLE/96.)
C!!!!!MULTIPLY NCPOUT BY THE NUMBER OF OUTPUT CYCLES DESIRED BETWEEN
C!!!!!PROGRAM EXITS. THIS DETERMINES THE INTERVAL FOR PLOT3D FILES.
            NEXIT = 24 * NCPOUT
            ACYC = AMOD(TINCR,CYCLE)
            DO 10 J=1,4
               IF (ABS(TINCR - CYCLE*J) .LT. 0.5 .AND. (ITN .GT.
     1            NCPOUT)) ACYC = 96. / (NEXIT / NCPOUT) * NEXIT
   10 CONTINUE
            NBCYC = NINT(ACYC)
            IF (NBCYC/NCPOUT * NCPOUT .EQ. NBCYC) THEN
               INDEX = NBCYC / NCPOUT
               ALPHA(INDEX) = ALFAD
C     ****
C
               WRITE (6,19)
               WRITE (6,33) ITN
               WRITE (6,3500) ALFAD
               CALL CPPLOT(ITEL,ITEU,AMINF,Y(1,1),Z(1,1),PSUR)
C
C!!!!!STORE CURRENT PRESSURE COEFFICIENTS. ADDED STATEMENTS BY JFV.
               DO 20 J=1,97
                  CPTH(J,INDEX) = PSUR(J+ITEL-1)
   20 CONTINUE
C
C!!!!!
               IF (FULOUT .GE. 0.) WRITE(6,12) (I,CF(I),I=1,IMAX)
               CALL LOAD(PSUR,CL,CD,CM,ALFAS)
               WRITE (6,3000) CL , CD , CM
C
C!!!!!FOR AUTOMATIC PROGRAM EXIT DURING FINAL OUTPUT (JFV)
C
               IF ((NBCYC/NEXIT * NEXIT .EQ. NBCYC) .AND. (ITN .GT.
     1            2 * NCPOUT)) GO TO 1001
               WRITE (6,3002)
C     ****
C
            END IF
         END IF
C
C
 1000 CONTINUE
C!!!!!
 1001 CONTINUE
      WRITE (8) TIME,Q1,Q2,Q3,Q4
C!!!!!SAVE PRESSURE COEFFICIENT DATA (JFV).
      IF (FULOUT .GE. 0.) THEN
         WRITE (18,1) TITLE
         WRITE (18) AMPLTD,STMEAN,OMS,XTH,ALPHA,CPTH
C
C!!!!!CREATE PLOT3D FILES. FEATURE ADDED BY JFV.
C!!!!!DO NOT USE WHEN TROUBLESHOOTING (ABOVE)
C
         WRITE(11) IMAX, KMAX
         WRITE(11) ((X(I,J), I=1,IMAX), J=1,KMAX),
```

```
     1              ((Z(I,J), I=1,IMAX), J=1,KMAX)
          WRITE(12) IMAX, KMAX
          WRITE(12) AMINF, ALFAD, REYREF, TIME
          WRITE(12) ((Q1(I,J), I=1,IMAX), J=1,KMAX),
     1              ((Q2(I,J), I=1,IMAX), J=1,KMAX),
     2              ((Q3(I,J), I=1,IMAX), J=1,KMAX),
     3              ((Q4(I,J), I=1,IMAX), J=1,KMAX)
      END IF
C
C       PRINT OUT VELOCITY PROFILE
C
      DO 4000 I = 1 , IMAX , 2
         S = 0.
         DO 4000 K = 2 , 10
         S = S + SQRT((X(I,K)-X(I,K-1))**2+(Z(I,K)-Z(I,K-1))**2)
         ED = CMU(I,K)  / DT
         UTOT = SQRT(Q2(I,K)**2 + Q3(I,K)**2)/Q1(I,K)
C!!!!!!IF PRINTED OUTPUT IS DESIRED
C         WRITE (6,2002) I , K,Q2(I,K) , Q3(I,K) , ED , UTOT,S
 4000 CONTINUE
      STOP
    1 FORMAT(15A4)
    2 FORMAT(2I5,7F10.0)
    3 FORMAT(/,5X,5HIMAX=,I10,//,5X,5HKMAX=,I10,/,5X,5H  DT=,F20.8,
     1/,5X,5H  WW=,F20.8,/,5X,5HALFA=,F20.8,/,5X,6HALFA1=,F20.8,
     2/,5X,6HALFAI=,F20.8,/,5X,7HREDFRE=,F20.8,/,5X,6HAMINF=,F20.8)
    4 FORMAT(1H1,5X,15A4)
   12 FORMAT(8(I4,F10.4))
   19 FORMAT(/)
   22 FORMAT(2F10.6,I5)
   33 FORMAT(5X,5HISTP=,I5)
 2002 FORMAT(2I5,5E14.6)
 3000 FORMAT(5X,3HCL=,F10.4,5X,3HCD=,F10.4,5X,3HCM=,F10.4)
 3002 FORMAT(//,4X,'DRMAX',11X,'DUMAX',11X,'DVMAX',11X,'DEMAX',10X,
     1'IR',3X,'KR')
 3500 FORMAT(/,5X,5HALFA=,F10.4)
 3700 FORMAT(5X,7HREYREF=,F13.4)
      END
      SUBROUTINE AMAT1(K,IMX1,XIX,XIZ,XIT)
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      COMMON/PERTR/DQ1(161,41),DQ2(161,41),DQ3(161,41),DQ4(161,41)
      COMMON/AM/A(4,4,161)
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      DIMENSION XIT(161,41),XIX(161,41),XIZ(161,41)
      REAL K0,K1,K2
C
C***  AMAT1 COMPUTES THE COEFFICIENT MATRIX DE/DQ DURING XI SWEEP
C
      GM1       = GAMMA - 1.
      DO 1000 I = 2 , IMX1
      K0        = XIT(I,K)
      K1        = XIX(I,K)
      K2        = XIZ(I,K)
      U         = Q2(I,K) / Q1(I,K)
      W         = Q3(I,K) / Q1(I,K)
      EBYR      = Q4(I,K) / Q1(I,K)
      PHI2      = 0.5 * GM1 * (U * U + W * W)
      THETA     = K1 * U + K2 * W
      A(1,1,I)  = K0
      A(1,2,I)  = K1
```

```
      A(1,3,I)  = K2
      A(1,4,I)  = 0
      A(2,1,I)  = K1 * PHI2 - U * THETA
      A(2,2,I)  = K0 + THETA - K1 * (GM1 - 1.) * U
      A(2,3,I)  = K2 * U - GM1 * K1 * W
      A(2,4,I)  = K1 * GM1
      A(3,1,I)  = K2 * PHI2 - W * THETA
      A(3,2,I)  = K1 * W - K2 * GM1 * U
      A(3,3,I)  = K0 + THETA - K2 * (GM1 - 1.) * W
      A(3,4,I)  = K2 * GM1
      A(4,1,I)  = THETA * (2. * PHI2 - GAMMA * EBYR)
      A(4,2,I)  = K1 * (GAMMA * EBYR - PHI2) - GM1 * U * THETA
      A(4,3,I)  = K2 * (GAMMA * EBYR - PHI2) - GM1 * W * THETA
      A(4,4,I)  = K0 + GAMMA * THETA
 1000 CONTINUE
      RETURN
      END
      SUBROUTINE AMAT2(I,KMX1,ZETAX,ZETAZ,ZETAT)
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      COMMON/PERTR/DQ1(161,41),DQ2(161,41),DQ3(161,41),DQ4(161,41)
      COMMON/AM/A(4,4,161)
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      DIMENSION ZETAX(161,41),ZETAZ(161,41),ZETAT(161,41)
      REAL K0,K1,K2
C
C***  AMAT2 COMPUTES THE COEFFICIENT MATRIX DF/DQ DURING ETA SWEEP
C
      GM1       = GAMMA - 1.
      DO 1000 K = 2 , KMX1
      K0        = ZETAT(I,K)
      K1        = ZETAX(I,K)
      K2        = ZETAZ(I,K)
      U         = Q2(I,K) / Q1(I,K)
      W         = Q3(I,K) / Q1(I,K)
      EBYR      = Q4(I,K) / Q1(I,K)
      PHI2      = 0.5 * GM1 * (U * U + W * W)
      THETA     = K1 * U + K2 * W
      A(1,1,K)  = K0
      A(1,2,K)  = K1
      A(1,3,K)  = K2
      A(1,4,K)  = 0
      A(2,1,K)  = K1 * PHI2 -U * THETA
      A(2,2,K)  = K0 + THETA - K1 * (GM1-1.) * U
      A(2,3,K)  = K2 * U - GM1 * K1 * W
      A(2,4,K)  = K1 * GM1
      A(3,1,K)  = K2 * PHI2 - W * THETA
      A(3,2,K)  = K1 * W - K2 * GM1 * U
      A(3,3,K)  = K0 + THETA -K2 * (GM1-1.) * W
      A(3,4,K)  = K2 * GM1
      A(4,1,K)  = THETA * (2. * PHI2 - GAMMA * EBYR)
      A(4,2,K)  = K1 * (GAMMA * EBYR - PHI2) - GM1 * U * THETA
      A(4,3,K)  = K2 * (GAMMA * EBYR - PHI2) - GM1 * W * THETA
      A(4,4,K)  = K0 + GAMMA * THETA
 1000 CONTINUE
      RETURN
      END
      SUBROUTINE SLPS(ITN,OMEGA,DALFA)
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      COMMON/PERTR/DQ1(161,41),DQ2(161,41),DQ3(161,41),DQ4(161,41)
      COMMON/AM/A(4,4,161)
```

```
      COMMON/TRID/DD(4,4,161,41),MM(4,4,161,41),EE(4,4,161,41)
     1,GG(4,161,41)
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/GRID/YACOB(161,41)
      COMMON/DAMP/WW,WWI
      COMMON/MTRIX/ XIX(161,41),XIZ(161,41),ZETAX(161,41),ZETAZ(161,41)
     1 ,XIT(161,41),ZETAT(161,41)
      REAL MM
      DIMENSION DELTAT(161,41)
C
C***  SUBROUTINE SLPS DOES THE BULK OF THE WORK  FOR THE ADI ALGORITHM.
C***  IT CALLS FLUX AND COMPUTES RIGHT HAND SIDE DURING THE TWO SWEEPS,
C***  ASSEMBLES THE COEFFICIENT MARICES, ADDS IMPLICIT AND EXPLICIT
C***  DISSIPATION AND CALLS THE TRIDIAGONAL SOLVER TO OBTAIN THE FINAL
C***  SOLUTION.
C!!!!!SET VALUE OF IMPLICIT DAMPING COEFFICIENT
      WWI = 20.0 * WW
      IM1 = IMAX - 1
      IM2 = IMAX - 2
      KM1 = KMAX - 1
      KM2 = KMAX - 2
      IF(ITN.EQ.1) THEN
C!!!!!LOCAL TIME STEP OPTION FOR STEADY STATE CALCULATIONS
      IF(REDFRE.LT.0.001) THEN
      DO 777 K = 2 , KMAX - 1
      DO 777 I = 2 , IMAX - 1
      DELTAT(I,K) = 0.5 / (1. + SQRT(ABS(YACOB(I,K))))
  777 CONTINUE
      ELSE
      DO 778 K = 2 , KMAX - 1
      DO 778 I = 2 , IMAX - 1
  778 DELTAT(I,K) = 1.0
      END IF
      END IF
C
C***  THE DISSIPATION TERMS ARE CONSTRUCTED AND STORED IN THE ARRAYS DQ1,
C***  DQ2,DQ3 AND DQ4.
C
      CALL DISSIP
C
C***  THE RIGHT HAND SIDE AT KNOWN TIME LEVEL IS NOW COMPUTED AND ADDED
      CALL RESI(OMEGA)
C
C***  IF VISCOUS FLOW IS COMPUTED THE VISCOUS TERMS ARE ADDED TO DQ1 ETC. HERE.
C
      IF(REYREF.GT.0.) CALL STRESS(ITN,DALFA)
C***  I-SWEEP.
C
      DTH = DT * 0.5
      DTW = DT * WWI
      DO 3 K          = 2 , KM1
      CALL AMAT1(K,IMAX-1,XIX,XIZ,XIT)
      DO 4 I1         = 1 , 4
      DO 4 I2         = 1 , 4
      DO 5 I          = 2 , IMAX - 1
      EE(I1,I2,I-1,K) =   A(I1,I2,I+1) * DTH * DELTAT(I,K)
      DD(I1,I2,I-1,K) = - A(I1,I2,I-1) * DTH * DELTAT(I,K)
    5 CONTINUE
    4 CONTINUE
```

```
C
C***  IMPLICIT DAMPING ADDED HERE.
C
      DO 6 I1          = 1 , 4
      DO 6 I           = 2 , IMAX - 1
      DT1              = DTW / YACOB(I,K) * DELTAT(I,K)
      DD(I1,I1,I-1,K) = DD(I1,I1,I-1,K) - DT1 * YACOB(I-1,K)
      EE(I1,I1,I-1,K) = EE(I1,I1,I-1,K) - DT1 * YACOB(I+1,K)
      MM(I1,I1,I-1,K) = 1. + 2. * DTW * DELTAT(I,K)
    6 CONTINUE
      DO 990 I         = 2 , IMAX - 1
      GG(1,I-1,K)      = DQ1(I,K) * DELTAT(I,K)
      GG(2,I-1,K)      = DQ2(I,K) * DELTAT(I,K)
      GG(3,I-1,K)      = DQ3(I,K) * DELTAT(I,K)
      GG(4,I-1,K)      = DQ4(I,K) * DELTAT(I,K)
  990 CONTINUE
    3 CONTINUE
C
C     PERFORM BLOCK TRIDIAGONAL MATRIX INVERSION FOR THE ENTIRE PLANE
C
      CALL MATRX1(IMAX,KMAX)
      DO 991 K         = 2 , KMAX - 1
      DO 991 I         = 2 , IM1
      DQ1(I,K)         = GG(1,I-1,K)
      DQ2(I,K)         = GG(2,I-1,K)
      DQ3(I,K)         = GG(3,I-1,K)
      DQ4(I,K)         = GG(4,I-1,K)
  991 CONTINUE
C
C     K-SWEEP BEGINS HERE.
C
      DO 13 I          = 2 , IM1
      CALL AMAT2(I,KMAX-1,ZETAX,ZETAZ,ZETAT)
      DO 15 I1         = 1 , 4
      DO 15 I2         = 1 , 4
      DO 15 K          = 2 , KMAX - 1
      EE(I1,I2,I,K-1) = A(I1,I2,K+1)*DTH * DELTAT(I,K)
      DD(I1,I2,I,K-1) = -A(I1,I2,K-1)*DTH * DELTAT(I,K)
   15 CONTINUE
C
C***  SECOND ORDER DAMPING ADDED HERE.
C
      DO 16 I1         = 1 , 4
      DO 16 K          = 2 , KMAX - 1
      DT1              = DTW / YACOB(I,K) * DELTAT(I,K)
      DD(I1,I1,I,K-1) = DD(I1,I1,I,K-1) - DT1 * YACOB(I,K-1)
      EE(I1,I1,I,K-1) = EE(I1,I1,I,K-1) - DT1 * YACOB(I,K+1)
   16 MM(I1,I1,I,K-1) = 1. + 2. *  DTW * DELTAT(I,K)
      DO 17 K          = 2 , KMAX - 1
      GG(1,I,K-1)      = DQ1(I,K)
      GG(2,I,K-1)      = DQ2(I,K)
      GG(3,I,K-1)      = DQ3(I,K)
      GG(4,I,K-1)      = DQ4(I,K)
   17 CONTINUE
   13 CONTINUE
C
C     PERFORM BLOCK TRIDIAGONAL MATRIX INVERSION FOR THE ENTIRE PLANE
C
      CALL MATRX2(IMAX,KMAX)
      DO 18 I          = 2 , IMAX - 1
```

```
      DO 18 K          = 2 , KM1
      DQ1(I,K)         = GG(1,I,K-1)
      DQ2(I,K)         = GG(2,I,K-1)
      DQ3(I,K)         = GG(3,I,K-1)
      DQ4(I,K)         = GG(4,I,K-1)
   18 CONTINUE
C
C***  UPDATE FLOW VARIABLES AT INTERIOR POINTS.
  967 CONTINUE
      RMAX             = 0.
      RUMAX            = 0.
      RVMAX            = 0.
      EMAX             = 0.
      DO 995 K         = 2 , KM1
      DO 19  I         = 2 , IM1
      Q1(I,K)          = Q1(I,K) + DQ1(I,K) * YACOB(I,K)
      Q2(I,K)          = Q2(I,K) + DQ2(I,K) * YACOB(I,K)
      Q3(I,K)          = Q3(I,K) + DQ3(I,K) * YACOB(I,K)
      Q4(I,K)          = Q4(I,K) + DQ4(I,K) * YACOB(I,K)
   19 CONTINUE
C!!!!!DETERMINE WHERE IN FLOW FIELD DENSITY IS CHANGING MOST RAPIDLY
      DO 995 I         = 2 , IMAX - 1
      IF (RMAX.LT.ABS(DQ1(I,K)*YACOB(I,K))) THEN
      IR               = I
      KR               = K
      END IF
      RMAX             = AMAX1(RMAX,ABS(DQ1(I,K) * YACOB(I,K)))
      RUMAX            = AMAX1(RUMAX,ABS(DQ2(I,K) * YACOB(I,K)))
      RVMAX            = AMAX1(RVMAX,ABS(DQ3(I,K) * YACOB(I,K)))
      EMAX             = AMAX1(EMAX,ABS(DQ4(I,K) * YACOB(I,K)))
  995 CONTINUE
C       IF((ITN-1)/100*100.EQ.(ITN-1)) WRITE (6,3002)
      IF(ITN .EQ. 0) WRITE (6,3002)
C!!!!!SELECT INTERVAL AT WHICH OUTPUT OF RESIDUALS IS DESIRED
      IF((ITN-1)/10*10.EQ.(ITN-1)) WRITE (6,3001) RMAX,RUMAX,RVMAX,
     1EMAX,IR,KR
      RETURN
 3002 FORMAT(//,4X,'DRMAX',11X,'DUMAX',11X,'DVMAX',11X,'DEMAX',10X,
     1'IR',3X,'KR')
 3001 FORMAT(4(E14.8,2X),2I5)
      END
      SUBROUTINE MATRX1(IMAX,KMAX)
      COMMON/TRID/DD(4,4,161,41),MM(4,4,161,41),EE(4,4,161,41),
     1GG(4,161,41)
      COMMON/SCRAT/A(4,4,161),HH(4,4,161,41),C(4,5,161)
      REAL MM
      REAL L11,L21,L31,L41,L22,L32,L42,L33,L43,L44
     2,L1I,L2I,L3I,L4I
C
C     THIS SUBROUTINE PERFORMS THE BLOCK TRIDIAGONAL MATRIX IVERSION FOR
C     AN ENTIRE  PLANE DURING THE XI- SWEEP
C
      DO 1 I1 = 1, 4
      DO 1 K = 2 , KMAX - 1
      AI = 1. / MM(1,1,1,K)
      GG(I1,1,K) = GG(I1,1,K) * AI
      HH(I1,1,1,K) = EE(I1,1,1,K) * AI
      HH(I1,2,1,K) = EE(I1,2,1,K) * AI
      HH(I1,3,1,K) = EE(I1,3,1,K) * AI
      HH(I1,4,1,K) = EE(I1,4,1,K) * AI
```

```
    1 CONTINUE
C
      DO 1000 I = 2 , IMAX - 2
      DO 5    I1= 1 , 4
      DO 2    K = 2 , KMAX - 1
      C(I1,1,K)   = GG(I1,I,K) - DD(I1,1,I,K) * GG(1,I-1,K)
     1                         - DD(I1,2,I,K) * GG(2,I-1,K)
     2                         - DD(I1,3,I,K) * GG(3,I-1,K)
     3                         - DD(I1,4,I,K) * GG(4,I-1,K)
    2 CONTINUE
      DO 5   I2 = 1 , 4
      DO 5   K  = 2 , KMAX - 1
      A(I1,I2,K)= MM(I1,I2,I,K) - DD(I1,1,I,K) * MM(1,I2,I-1,K)
     1                          - DD(I1,2,I,K) * MM(2,I2,I-1,K)
     2                          - DD(I1,3,I,K) * MM(3,I2,I-1,K)
     3                          - DD(I1,4,I,K) * MM(4,I2,I-1,K)
      C(I1,I2+1,K)= EE(I1,I2,I,K)
    5 CONTINUE
      DO 3 K = 2 , KMAX - 1
      L11 = A(1,1,K)
      L1I = 1. / L11
      U12 = A(1,2,K) * L1I
      U13 = A(1,3,K) * L1I
      U14 = A(1,4,K) * L1I
      L21 = A(2,1,K)
      L31 = A(3,1,K)
      L41 = A(4,1,K)
      L22 = A(2,2,K) - L21 * U12
      L2I = 1. / L22
      U23 = (A(2,3,K) - L21 * U13) * L2I
      U24 = (A(2,4,K) - L21 * U14) * L2I
      L32 = A(3,2,K) - L31 * U12
      L42 = A(4,2,K) - L41 * U12
      L33 = A(3,3,K) - L31 * U13 - L32 * U23
      L3I = 1. / L33
      U34 = (A(3,4,K) - L31 * U14 - L32 * U24) * L3I
      L43 = A(4,3,K) - L41 * U13 - L42 * U23
      L44 = A(4,4,K) - L41 * U14 - L42 * U24 - L43 * U34
      L4I = 1. / L44
      C(1,1,K) = C(1,1,K) * L1I
      C(2,1,K) = (C(2,1,K) - L21 * C(1,1,K)) * L2I
      C(3,1,K) = (C(3,1,K) - L31 * C(1,1,K)
     1                     - L32 * C(2,1,K)) * L3I
      C(4,1,K) = (C(4,1,K) - L41 * C(1,1,K) - L42 * C(2,1,K)
     1                                      - L43 * C(3,1,K)) * L4I
      C(1,2,K) = C(1,2,K) * L1I
      C(2,2,K) = (C(2,2,K) - L21 * C(1,2,K)) * L2I
      C(3,2,K) = (C(3,2,K) - L31 * C(1,2,K)
     1                     - L32 * C(2,2,K)) * L3I
      C(4,2,K) = (C(4,2,K) - L41 * C(1,2,K) - L42 * C(2,2,K)
     1                                      - L43 * C(3,2,K)) * L4I
      C(1,3,K) = C(1,3,K) * L1I
      C(2,3,K) = (C(2,3,K) - L21 * C(1,3,K)) * L2I
      C(3,3,K) = (C(3,3,K) - L31 * C(1,3,K)
     1                     - L32 * C(2,3,K)) * L3I
      C(4,3,K) = (C(4,3,K) - L41 * C(1,3,K) - L42 * C(2,3,K)
     1                                      - L43 * C(3,3,K)) * L4I
      C(1,4,K) = C(1,4,K) * L1I
      C(2,4,K) = (C(2,4,K) - L21 * C(1,4,K)) * L2I
      C(3,4,K) = (C(3,4,K) - L31 * C(1,4,K)
```

```
     1                        - L32 * C(2,4,K)) * L3I
      C(4,4,K) = (C(4,4,K) - L41 * C(1,4,K) - L42 * C(2,4,K)
     1                             - L43 * C(3,4,K)) * L4I
      C(1,5,K) =  C(1,5,K) * L1I
      C(2,5,K) = (C(2,5,K) - L21 * C(1,5,K)) * L2I
      C(3,5,K) = (C(3,5,K) - L31 * C(1,5,K)
     1                        - L32 * C(2,5,K)) * L3I
      C(4,5,K) = (C(4,5,K) - L41 * C(1,5,K) - L42 * C(2,5,K)
     1                             - L43 * C(3,5,K)) * L4I
      C(3,1,K) = C(3,1,K) - U34 * C(4,1,K)
      C(2,1,K) = C(2,1,K) - U24 * C(4,1,K)
     1                    - U23 * C(3,1,K)
      C(1,1,K) = C(1,1,K) - U14 * C(4,1,K)
     1                    - U13 * C(3,1,K) - U12 * C(2,1,K)
      C(3,2,K) = C(3,2,K) - U34 * C(4,2,K)
      C(2,2,K) = C(2,2,K) - U24 * C(4,2,K)
     1                    - U23 * C(3,2,K)
      C(1,2,K) = C(1,2,K) - U14 * C(4,2,K)
     1                    - U13 * C(3,2,K) - U12 * C(2,2,K)
      C(3,3,K) = C(3,3,K) - U34 * C(4,3,K)
      C(2,3,K) = C(2,3,K) - U24 * C(4,3,K)
     1                    - U23 * C(3,3,K)
      C(1,3,K) = C(1,3,K) - U14 * C(4,3,K)
     1                    - U13 * C(3,3,K) - U12 * C(2,3,K)
      C(3,4,K) = C(3,4,K) - U34 * C(4,4,K)
      C(2,4,K) = C(2,4,K) - U24 * C(4,4,K)
     1                    - U23 * C(3,4,K)
      C(1,4,K) = C(1,4,K) - U14 * C(4,4,K)
     1                    - U13 * C(3,4,K) - U12 * C(2,4,K)
      C(3,5,K) = C(3,5,K) - U34 * C(4,5,K)
      C(2,5,K) = C(2,5,K) - U24 * C(4,5,K)
     1                    - U23 * C(3,5,K)
      C(1,5,K) = C(1,5,K) - U14 * C(4,5,K)
     1                    - U13 * C(3,5,K) - U12 * C(2,5,K)
    3 CONTINUE
C
      DO 6 I1       = 1 , 4
      DO 9 K        = 2 , KMAX - 1
    9 GG(I1,I,K)    = C(I1,1,K)
      DO 6 I2       = 1 , 4
      DO 6 K        = 2 , KMAX - 1
      HH(I1,I2,I,K) = C(I1,I2+1,K)
    6 CONTINUE
 1000 CONTINUE
C
C     BACKWARD SUBSTITUTION
C
      DO 7 I     = IMAX - 3, 1 , - 1
      DO 7 I1    = 1 , 4
      DO 7 K     = 2 , KMAX - 1
      GG(I1,I,K) = GG(I1,I,K) - HH(I1,1,I,K) * GG(1,I+1,K)
     1                        - HH(I1,2,I,K) * GG(2,I+1,K)
     2                        - HH(I1,3,I,K) * GG(3,I+1,K)
     3                        - HH(I1,4,I,K) * GG(4,I+1,K)
    7 CONTINUE
      RETURN
      END
      SUBROUTINE MATRX2(IMAX,KMAX)
      COMMON/TRID/DD(4,4,161,41),MM(4,4,161,41),EE(4,4,161,41),
     1GG(4,161,41)
```

```
      COMMON/SCRAT/A(4,4,161),HH(4,4,161,41),C(4,5,161)
      REAL MM
      REAL L11,L21,L31,L41,L22,L32,L42,L33,L43,L44
     2,L1I,L2I,L3I,L4I
C
C     THIS SUBROUTINE PERFORMS THE BLOCK TRIDIAGONAL MATRIX IVERSION FOR
C     AN ENTIRE J=CONSTANT PLANE DURING THE ZETA- SWEEP
C
      DO 1 I1      = 1, 4
      DO 1 I       = 2 , IMAX - 1
      AI           = 1. / MM(1,1,I,1)
      GG(I1,I,1)   = GG(I1,I,1) * AI
      HH(I1,1,I,1) = EE(I1,1,I,1) * AI
      HH(I1,2,I,1) = EE(I1,2,I,1) * AI
      HH(I1,3,I,1) = EE(I1,3,I,1) * AI
      HH(I1,4,I,1) = EE(I1,4,I,1) * AI
    1 CONTINUE
C
      DO 1000 K = 2 , KMAX - 2
      DO 5    I1= 1 , 4
      DO 2    I = 2 , IMAX - 1
      C(I1,1,I) = GG(I1,I,K) - DD(I1,1,I,K) * GG(1,I,K-1)
     1                       - DD(I1,2,I,K) * GG(2,I,K-1)
     2                       - DD(I1,3,I,K) * GG(3,I,K-1)
     3                       - DD(I1,4,I,K) * GG(4,I,K-1)
    2 CONTINUE
      DO 5   I2 = 1 , 4
      DO 5   I  = 2 , IMAX - 1
      A(I1,I2,I)= MM(I1,I2,I,K) - DD(I1,1,I,K) * HH(1,I2,I,K-1)
     1                          - DD(I1,2,I,K) * HH(2,I2,I,K-1)
     2                          - DD(I1,3,I,K) * HH(3,I2,I,K-1)
     3                          - DD(I1,4,I,K) * HH(4,I2,I,K-1)
      C(I1,I2+1,I)= EE(I1,I2,I,K)
    5 CONTINUE
      DO 3 I = 2 , IMAX - 1
      L11 = A(1,1,I)
      L1I = 1. / L11
      U12 = A(1,2,I) * L1I
      U13 = A(1,3,I) * L1I
      U14 = A(1,4,I) * L1I
      L21 = A(2,1,I)
      L31 = A(3,1,I)
      L41 = A(4,1,I)
      L22 = A(2,2,I) - L21 * U12
      L2I = 1. / L22
      U23 = (A(2,3,I) - L21 * U13) * L2I
      U24 = (A(2,4,I) - L21 * U14) * L2I
      L32 = A(3,2,I) - L31 * U12
      L42 = A(4,2,I) - L41 * U12
      L33 = A(3,3,I) - L31 * U13 - L32 * U23
      L3I = 1. / L33
      U34 = (A(3,4,I) - L31 * U14 - L32 * U24) * L3I
      L43 = A(4,3,I) - L41 * U13 - L42 * U23
      L44 = A(4,4,I) - L41 * U14 - L42 * U24 - L43 * U34
      L4I = 1. / L44
      C(1,1,I) = C(1,1,I) * L1I
      C(2,1,I) = (C(2,1,I) - L21 * C(1,1,I)) * L2I
      C(3,1,I) = (C(3,1,I) - L31 * C(1,1,I)
     1                     - L32 * C(2,1,I)) * L3I
      C(4,1,I) = (C(4,1,I) - L41 * C(1,1,I) - L42 * C(2,1,I)
```

```
     1                                  - L43 * C(3,1,I)) * L4I
      C(1,2,I) = C(1,2,I) * L1I
      C(2,2,I) = (C(2,2,I) - L21 * C(1,2,I)) * L2I
      C(3,2,I) = (C(3,2,I) - L31 * C(1,2,I)
     1                     - L32 * C(2,2,I)) * L3I
      C(4,2,I) = (C(4,2,I) - L41 * C(1,2,I) - L42 * C(2,2,I)
     1                                  - L43 * C(3,2,I)) * L4I
      C(1,3,I) = C(1,3,I) * L1I
      C(2,3,I) = (C(2,3,I) - L21 * C(1,3,I)) * L2I
      C(3,3,I) = (C(3,3,I) - L31 * C(1,3,I)
     1                     - L32 * C(2,3,I)) * L3I
      C(4,3,I) = (C(4,3,I) - L41 * C(1,3,I) - L42 * C(2,3,I)
     1                                  - L43 * C(3,3,I)) * L4I
      C(1,4,I) = C(1,4,I) * L1I
      C(2,4,I) = (C(2,4,I) - L21 * C(1,4,I)) * L2I
      C(3,4,I) = (C(3,4,I) - L31 * C(1,4,I)
     1                     - L32 * C(2,4,I)) * L3I
      C(4,4,I) = (C(4,4,I) - L41 * C(1,4,I) - L42 * C(2,4,I)
     1                                  - L43 * C(3,4,I)) * L4I
      C(1,5,I) = C(1,5,I) * L1I
      C(2,5,I) = (C(2,5,I) - L21 * C(1,5,I)) * L2I
      C(3,5,I) = (C(3,5,I) - L31 * C(1,5,I)
     1                     - L32 * C(2,5,I)) * L3I
      C(4,5,I) = (C(4,5,I) - L41 * C(1,5,I) - L42 * C(2,5,I)
     1                                  - L43 * C(3,5,I)) * L4I
      C(3,1,I) = C(3,1,I) - U34 * C(4,1,I)
      C(2,1,I) = C(2,1,I) - U24 * C(4,1,I)
     1                     - U23 * C(3,1,I)
      C(1,1,I) = C(1,1,I) - U14 * C(4,1,I)
     1                     - U13 * C(3,1,I) - U12 * C(2,1,I)
      C(3,2,I) = C(3,2,I) - U34 * C(4,2,I)
      C(2,2,I) = C(2,2,I) - U24 * C(4,2,I)
     1                     - U23 * C(3,2,I)
      C(1,2,I) = C(1,2,I) - U14 * C(4,2,I)
     1                     - U13 * C(3,2,I) - U12 * C(2,2,I)
      C(3,3,I) = C(3,3,I) - U34 * C(4,3,I)
      C(2,3,I) = C(2,3,I) - U24 * C(4,3,I)
     1                     - U23 * C(3,3,I)
      C(1,3,I) = C(1,3,I) - U14 * C(4,3,I)
     1                     - U13 * C(3,3,I) - U12 * C(2,3,I)
      C(3,4,I) = C(3,4,I) - U34 * C(4,4,I)
      C(2,4,I) = C(2,4,I) - U24 * C(4,4,I)
     1                     - U23 * C(3,4,I)
      C(1,4,I) = C(1,4,I) - U14 * C(4,4,I)
     1                     - U13 * C(3,4,I) - U12 * C(2,4,I)
      C(3,5,I) = C(3,5,I) - U34 * C(4,5,I)
      C(2,5,I) = C(2,5,I) - U24 * C(4,5,I)
     1                     - U23 * C(3,5,I)
      C(1,5,I) = C(1,5,I) - U14 * C(4,5,I)
     1                     - U13 * C(3,5,I) - U12 * C(2,5,I)
    3 CONTINUE
C
      DO 6 I1       = 1 , 4
      DO 9 I        = 2 , IMAX - 1
    9 GG(I1,I,K)    = C(I1,1,I)
      DO 6 I2       = 1 , 4
      DO 6 I        = 2 , IMAX - 1
      HH(I1,I2,I,K) = C(I1,I2+1,I)
    6 CONTINUE
 1000 CONTINUE
```

```
C
C     BACKWARD SUBSTITUTION
C
      DO 7 K      = KMAX - 3, 1 , - 1
      DO 7 I1     = 1 , 4
      DO 7 I      = 2 , IMAX - 1
      GG(I1,I,K) = GG(I1,I,K) - HH(I1,1,I,K) * GG(1,I,K+1)
     1                        - HH(I1,2,I,K) * GG(2,I,K+1)
     2                        - HH(I1,3,I,K) * GG(3,I,K+1)
     3                        - HH(I1,4,I,K) * GG(4,I,K+1)
    7 CONTINUE
      RETURN
      END
      SUBROUTINE METRIC
      COMMON/FIX/OMEGA
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/GRID1/X(161,41),Z(161,41)
      COMMON/GRID/YACOB(161,41)
      COMMON/MTRIX/XIX(161,41),XIZ(161,41),ZETAX(161,41),ZETAZ(161,41),
     1XIT(161,41),ZETAT(161,41)
C
C***  SUBROUTINE METRIC COMPUTES THE METRICS IN BOTH DIRECTIONS AND
C     THE UNSTEADY COEFFICIENTS ETAT, ETC.
C
      DO 1000 K = 1 , KMAX
      DO 1000 I = 1 , IMAX
      XTAU = OMEGA * Z(I,K)
      YTAU = OMEGA * (-X(I,K))
C***  PRESENT SET UP IS FOR FLOW PAST AN  AIRFOIL.
C
C!!!!!CENTRAL DIFFERENCES AT INTERIOR POINTS, TWO-POINT ONE-SIDED
C!!!!!DIFFERENCES DOWNSTREAM, THREE-POINT AT OTHER OUTER BOUNDARIES
      IF(I.EQ.1.OR.I.EQ.IMAX) GO TO 10
      XXI = .5 * (X(I+1,K)-X(I-1,K))
      ZXI = .5 * (Z(I+1,K)-Z(I-1,K))
      GO TO 15
   10 IF(I.EQ.IMAX) GO TO 16
      XXI = 1.0 * (X(2,K) - X(1,K))
      ZXI = 1.0 * (Z(2,K) - Z(1,K))
      GO TO 15
   16 XXI = 1.0 * (X(IMAX,K) - X(IMAX-1,K))
      ZXI = 1.0 * (Z(IMAX,K) - Z(IMAX-1,K))
   15 CONTINUE
      IF(K.EQ.1.OR.K.EQ.KMAX) GO TO 17
      XZET = .5 *(X(I,K+1)-X(I,K-1))
      ZZET = .5 *(Z(I,K+1)-Z(I,K-1))
      GO TO 20
  17  IF(K.EQ.KMAX) GO TO 18
      XZET = 2. * X(I,2)-1.5 * X(I,1) - .5 * X(I,3)
      ZZET = 2. * Z(I,2) - 1.5 * Z(I,1) - .5 * Z(I,3)
      GO TO 20
   18 XZET = 1.5 * X(I,KMAX)-2.* X(I,KMAX-1)+.5*X(I,KMAX-2)
      ZZET = 1.5 * Z(I,KMAX)-2.* Z(I,KMAX-1)+.5*Z(I,KMAX-2)
   20 CONTINUE
C!!!!!COMPUTE JACOBIAN
      YACOBI = XXI * ZZET - XZET * ZXI
      YACOB(I,K) = 1. / YACOBI
      XIX(I,K) = ZZET * YACOB(I,K)
      XIZ(I,K) = -XZET * YACOB(I,K)
      XTAU = OMEGA * Z(I,K)
```

```
      YTAU = - OMEGA * X(I,K)
      XIT(I,K) = - XIX(I,K) * XTAU - XIZ(I,K) * YTAU
      ZETAX(I,K) = -ZXI * YACOB(I,K)
      ZETAZ(I,K) = XXI * YACOB(I,K)
      ZETAT(I,K) = - ZETAX(I,K) * XTAU - ZETAZ(I,K) * YTAU
 1000 CONTINUE
      RETURN
      END
      SUBROUTINE DISSIP
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      COMMON/PERTR/DQ1(161,41),DQ2(161,41),DQ3(161,41),DQ4(161,41)
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/GRID/YACOB(161,41)
      COMMON/DAMP/WW,WWI
      DIMENSION P(161),EPS(161),DIS1(161,4),DIS2(161,4)
C
C     THIS SUBROUTINE ADDS THE FOURTH ORDER DISSIPATION TERMS TO THE
C     RIGHT HAND SIDE
C
      IM1 = IMAX - 1
      KM1 = KMAX - 1
      IM2 = IMAX - 2
      KM2 = KMAX - 2
C
      DO 10 K=2 , KM1
C     COMPUTE SWTICHING FUNCTION BASED ON SECOND DERIVATIVE OF PRESSURE
      DO 1 I = 1 , IMAX
    1 P(I) = .4 * (Q4(I,K)-(Q2(I,K)**2+Q3(I,K)**2)/(2.*Q1(I,K)))
      DO 2 I =1 , IMAX
      IP2 = I + 2
      IF(I.EQ.IM1) IP2 = IMAX
      IM2 = I - 2
      IF(I.EQ.2) IM2 = 1
      IP1 = I + 1
      IF(I.EQ.IMAX) IP1 = IMAX
      IM = I - 1
      IF(I.EQ.1) IM = 1
      IF(I.EQ.1) IM2 = 1
      IF(I.EQ.IMAX) IP2 = IMAX
      EPS(I) = ABS(P(IP1)-2.*P(I)+P(IM))/ABS(P(IP1)+2.*P(I)+P(IM))
C     VOL = 2.  /(YACOB(I,K)+YACOB(IP1,K))
      VOL = 1.
      DIS1(I,1) =  (Q1(IP1,K)-Q1(I,K))*VOL
      DIS1(I,2) =  (Q2(IP1,K)-Q2(I,K))*VOL
      DIS1(I,3) =  (Q3(IP1,K)-Q3(I,K))*VOL
      HP1 = Q4(IP1,K)+P(IP1)
      HP = Q4(I,K)+P(I)
      HM1 = Q4(IM,K) + P(IM)
      HP2 = Q4(IP2,K) + P(IP2)
      HPM = Q4(IM,K)+P(IM)
      DIS1(I,4) =  (HP1-HP)*VOL
      DIS2(I,1) = -(Q1(IP2,K)-3.*(Q1(IP1,K)-Q1(I,K))-Q1(IM,K))*VOL
      DIS2(I,2) = -(Q2(IP2,K)-3.*(Q2(IP1,K)-Q2(I,K))-Q2(IM,K))*VOL
      DIS2(I,3) = -(Q3(IP2,K)-3.*(Q3(IP1,K)-Q3(I,K))-Q3(IM,K))*VOL
      DIS2(I,4) = -(HP2-3.*(HP1-HP)-HPM)*VOL
    2 CONTINUE
      DO 15 I = 1 , IM1
      D2P = AMAX1(EPS(I),EPS(I+1))
      C22 = 60. * D2P
      C11 = AMAX1(0.0,(1.-C22))
```

```
      C22 = C22 * WW/YACOB(I,K) * DT
      C11 = C11 * WW/YACOB(I,K) * DT
C!!!!!SWITCH ON SECOND-ORDER AND SWITCH OFF FOURTH-ORDER DISSIPATION
C!!!!!IN VICINITY OF SHOCKS
      DIS1(I,1) = C11 * DIS2(I,1) + C22 * DIS1(I,1)
      DIS1(I,2) = C11 * DIS2(I,2) + C22 * DIS1(I,2)
      DIS1(I,3) = C11 * DIS2(I,3) + C22 * DIS1(I,3)
      DIS1(I,4) = C11 * DIS2(I,4) + C22 * DIS1(I,4)
   15 CONTINUE
      DO 16 I = 2 , IM1
      DQ1(I,K) = DIS1(I,1) - DIS1(I-1,1)
      DQ2(I,K) = DIS1(I,2) - DIS1(I-1,2)
      DQ3(I,K) = DIS1(I,3) - DIS1(I-1,3)
      DQ4(I,K) = DIS1(I,4) - DIS1(I-1,4)
   16 CONTINUE
   10 CONTINUE
C     K DIRECTION
C!!!!!FOURTH-ORDER DISSIPATION ONLY
      DO 30 I = 2 , IM1
      WT= 0.5 * DT * WW / YACOB(I,2)
      W3 = 0.5 * DT * WW / YACOB(I,KM1)
      DQ1(I,2) =WT* (Q1(I,1) - 2. * Q1(I,2) + Q1(I,3))
     1+DQ1(I,2)
      DQ2(I,2) =WT* (Q2(I,1) - 2. * Q2(I,2) + Q2(I,3))
     1+DQ2(I,2)
      DQ3(I,2) =WT* (Q3(I,1) - 2. * Q3(I,2) + Q3(I,3))
     1+DQ3(I,2)
      DQ4(I,2) =WT* (Q4(I,1) - 2. * Q4(I,2) + Q4(I,3))
     1+DQ4(I,2)
      WT= W3
      DQ1(I,KM1) =WT* (Q1(I,KM2) - 2. * Q1(I,KM1) + Q1(I,KMAX))
     1+DQ1(I,KM1)
      DQ2(I,KM1) =WT* (Q2(I,KM2) - 2. * Q2(I,KM1) + Q2(I,KMAX))
     1+DQ2(I,KM1)
      DQ3(I,KM1) =WT* (Q3(I,KM2) - 2. * Q3(I,KM1) + Q3(I,KMAX))
     1+DQ3(I,KM1)
      DQ4(I,KM1) =WT* (Q4(I,KM2) - 2. * Q4(I,KM1) + Q4(I,KMAX))
     1+DQ4(I,KM1)
      DO 35 K = 3 , KM2
      WT= - DT * WW / YACOB(I,K)
      DQ1(I,K) =WT* (Q1(I,K+2) - 4. * Q1(I,K+1) + 6. * Q1(
     1I,K) - 4. * Q1(I,K-1) + Q1(I,K-2))+DQ1(I,K)
      DQ2(I,K) =WT*(Q2(I,K+2) - 4. * Q2(I,K+1) + 6. * Q2(
     1I,K) - 4. * Q2(I,K-1) + Q2(I,K-2))+DQ2(I,K)
      DQ3(I,K) =WT*(Q3(I,K+2) - 4. * Q3(I,K+1) + 6. * Q3(
     1I,K) - 4. * Q3(I,K-1) + Q3(I,K-2))+DQ3(I,K)
      DQ4(I,K) =WT*(Q4(I,K+2) - 4. * Q4(I,K+1) + 6. * Q4(
     1I,K) - 4. * Q4(I,K-1) + Q4(I,K-2))+DQ4(I,K)
   35 CONTINUE
   30 CONTINUE
C
      RETURN
      END
      SUBROUTINE WALLBC
      COMMON/SURF/PSUR(161)
      COMMON/GRID1/X(161,41),Z(161,41)
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/GRID/YACOB(161,41)
      COMMON/DAMP/WW,WWI
```

```
      COMMON/MTRIX/XIX(161,41),XIZ(161,41),ZETAX(161,41),ZETAZ(161,41),
     1XIT(161,41),ZETAT(161,41)
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      DIMENSION C1(4)
      DIMENSION A(2,2),RHS(2)
C     APPLY BOUNDARY CONDITIONS ON THE CUT AND THE AIRFOIL SURFACE
      DO 9 I=ITEU,IMAX
      I1 = IMAX + 1 - I
      Q1(I,1) = .5 * (Q1(I,2)+Q1(I1,2))
      Q2(I,1) =.5*(Q2(I,2)+Q2(I1,2))
      Q3(I,1) = .5 * (Q3(I,2) + Q3(I1,2))
      Q4(I,1) = .5 * (Q4(I,2)+Q4(I1,2))
      Q1(I1,1)=Q1(I,1)
      Q2(I1,1)=Q2(I,1)
      Q3(I1,1)=Q3(I,1)
      Q4(I1,1)=Q4(I,1)
    9 CONTINUE
      DO 1 I=  ITEL , ITEU
      K = 3
      C1(1) = XIT(I,K)
      C1(2) = XIX(I,K)
      C1(3) = XIZ(I,K)
      UCON3 = (Q2(I,K)*C1(2)+Q3(I,K)*C1(3))
     1/Q1(I,K)
      K = 2
      C1(1) = XIT(I,K)
      C1(2) = XIX(I,K)
      C1(3) = XIZ(I,K)
      UCON2 = (Q2(I,K)*C1(2)+Q3(I,K)*C1(3))
     1/Q1(I,K)
      RHS(1) = 2. * UCON2 - UCON3 - XIT(I,1)
C     FOR VISCOUS FLOWS SET UCON TO ZERO ALSO
      IF(REYREF.GT.0.) RHS(1) = - XIT(I,1)
      A(1,1) = XIX(I,1)
      A(1,2) = XIZ(I,1)
      A(2,1) = ZETAX(I,1)
      A(2,2) = ZETAZ(I,1)
      RHS(2) = - ZETAT(I,1)
      TEMP1 = A(1,1)
      TEMP2 = A(1,2)
      TEMP3 = A(2,1)
      TEMP4 = A(2,2)
      DEN = 1. /(TEMP1 * TEMP4 - TEMP2 * TEMP3)
      A(1,1) = A(2,2) * DEN
      A(1,2) = - TEMP2 * DEN
      A(2,1) = - TEMP3 * DEN
      A(2,2) = TEMP1 * DEN
      Q1(I,1) = 2. * Q1(I,2) - Q1(I,3)
      Q2(I,1) = Q1(I,1)*(A(1,1)*RHS(1)+A(1,2)*RHS(2))
      Q3(I,1) = Q1(I,1) *(A(2,1)*RHS(1)+A(2,2)*RHS(2))
    1 CONTINUE
      DO 10 I=ITEL ,ITEU
      U2=Q2(I,2)/Q1(I,2)
      W2=Q3(I,2)/Q1(I,2)
      P2=(GAMMA-1.)*(Q4(I,2)-0.5*Q1(I,2)*(U2*U2+W2*W2))
      U3=Q2(I,3)/Q1(I,3)
      W3=Q3(I,3)/Q1(I,3)
      P3=(GAMMA-1.)*(Q4(I,3)-0.5*Q1(I,3)*(U3*U3+W3*W3))
      P1=(4.*P2-P3)/3.
      PSUR(I)=(GAMMA*P1-1.)/(.7*AMINF**2)
```

```
      U1=Q2(I,1)/Q1(I,1)
      W1=Q3(I,1)/Q1(I,1)
   10 Q4(I,1)=P1/(GAMMA-1.)+0.5*Q1(I,1)*(U1*U1+W1*W1)
      RETURN
      END
      SUBROUTINE STRESS(ITN,DALFA)
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/GRID1/X(161,41),Z(161,41)
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      COMMON/PERTR/DQ1(161,41),DQ2(161,41),DQ3(161,41),DQ4(161,41)
      COMMON/MUTUR/CMU(161,41)
      DIMENSION AA(161,41)
     1,RH1(161),RH2(161),RH3(161),RH4(161)
      COMMON/LOGIC/RSTRT,PITCH,PLUNGE
      LOGICAL RSTRT,PITCH,PLUNGE
      U(I,J) = Q2(I,J) / Q1(I,J)
      V(I,J) = Q3(I,J) / Q1(I,J)
C     THIS SUBROUTINE ADDS VISCOUS TERMS TO THE RIGHT HAND SIDE
      GOGM = GAMMA / (GAMMA - 1.)
      IF(ITN.GT.10.OR.(RSTRT)) CALL EDDY(DALFA)
C     COMPUTE U AND V
      KMAXM1 = KMAX - 1
      IMAXM1 = IMAX - 1
      PR = 1.
      DO 10 K = 1 , KMAX
      DO 10 I = 1 , IMAX
      E      = Q4(I,K) / Q1(I,K) - 0.5 * (U(I,K)**2+V(I,K)**2)
   10 AA(I,K) = GOGM * E
C
C     COMPUTE TXX,TXY AND VISCOUS DISSIPATION AT I - 1 / 2
C
      DO 30 K = 2 , KMAXM1
      DO 20 I = 2 , IMAX
      UXI = U(I,K) - U(I-1,K)
      VXI = V(I,K) - V(I-1,K)
      AXI = AA(I,K) - AA(I-1,K)
      UZET= .25*(U(I,K+1)-U(I,K-1)+U(I-1,K+1)-U(I-1,K-1))
      VZET= .25*(V(I,K+1)-V(I,K-1)+V(I-1,K+1)-V(I-1,K-1))
      AZET= .25*(AA(I,K+1)-AA(I,K-1)+AA(I-1,K+1)-AA(I-1,K-1))
      XXI = X(I,K) - X(I-1,K)
      ZXI = Z(I,K) - Z(I-1,K)
      XZET= .25 * (X(I,K+1)-X(I,K-1)+X(I-1,K+1)-X(I-1,K-1))
      ZZET= .25 * (Z(I,K+1)-Z(I,K-1)+Z(I-1,K+1)-Z(I-1,K-1))
      YAC = XXI * ZZET - ZXI * XZET
      YAC = 1. / YAC
      XIX = ZZET * YAC
      ZETAX= - ZXI * YAC
      XIZ = -XZET * YAC
      ZETAZ= XXI * YAC
      CNM = .5 * (CMU(I,K) + CMU(I-1,K))
      UX  = UXI * XIX + UZET * ZETAX
      VX  = VXI * XIX + VZET * ZETAX
      AX  = AXI * XIX + AZET * ZETAX
      UZ  = UXI * XIZ + UZET * ZETAZ
      VZ  = VXI * XIZ + VZET * ZETAZ
      AZ  = AXI * XIZ + AZET * ZETAZ
      TXX = -(-4. * UX + 2. * VZ) * CNM / 3.
      TXY = CNM * (UZ + VX)
      TYY = -CNM / 3. * (-4. * VZ + 2. * UX)
```

```
      R4  = ((U(I,K)+U(I-1,K))*TXX+(V(I,K)+V(I-1,K))*TXY)*0.5
     1     + CNM / PR/(GAMMA - 1.) * AX
      S4  = ((U(I,K)+U(I-1,K))*TXY+(V(I,K)+V(I-1,K))*TYY)*0.5
     1     + CNM / PR / (GAMMA - 1.) * AZ
C      DEBUG
C      TURN OFF ENRGY DISSIPATION AND DIFFUSION
      R4 = 0.
      S4 = 0.
      RH1(I) = 0.
      RH2(I) = (XIX * TXX + XIZ * TXY) / YAC
      RH3(I) = (XIX * TXY + XIZ * TYY) / YAC
   20 RH4(I) = (XIX * R4 + XIZ * S4) / YAC
      DO 30 I = 2 , IMAXM1
      DQ1(I,K) = DQ1(I,K) + RH1(I+1) - RH1(I)
      DQ2(I,K) = DQ2(I,K) + RH2(I+1) - RH2(I)
      DQ3(I,K) = DQ3(I,K) + RH3(I+1) - RH3(I)
      DQ4(I,K) = DQ4(I,K) + RH4(I+1) - RH4(I)
   30 CONTINUE
C     IN THE Z DIRECTION
      DO 70 I = 2 , IMAXM1
      DO 60 K = 2 , KMAX
      UXI = .25 * (U(I+1,K)-U(I-1,K)+U(I+1,K-1)-U(I-1,K-1))
      VXI = .25 * (V(I+1,K)-V(I-1,K)+V(I+1,K-1)-V(I-1,K-1))
      AXI = .25 * (AA(I+1,K)-AA(I-1,K)+AA(I+1,K-1)-AA(I-1,K-1))
      XXI = .25 * (X(I+1,K)-X(I-1,K)+X(I+1,K-1)-X(I-1,K-1))
      ZXI = .25 * (Z(I+1,K)-Z(I-1,K)+Z(I+1,K-1)-Z(I-1,K-1))
      UZET = U(I,K) - U(I,K-1)
      VZET = V(I,K) - V(I,K-1)
      AZET = AA(I,K) - AA(I,K-1)
      XZET = X(I,K) - X(I,K-1)
      ZZET = Z(I,K) - Z(I,K-1)
      YAC = XXI * ZZET - ZXI * XZET
      YAC = 1. / YAC
      XIX = ZZET * YAC
      ZETAX= - ZXI * YAC
      XIZ = -XZET * YAC
      ZETAZ= XXI * YAC
      CNM = .5 * (CMU(I,K) + CMU(I,K-1))
      UX  = UXI * XIX + UZET * ZETAX
      VX  = VXI * XIX + VZET * ZETAX
      AX  = AXI * XIX + AZET * ZETAX
      UZ  = UXI * XIZ + UZET * ZETAZ
      VZ  = VXI * XIZ + VZET * ZETAZ
      AZ  = AXI * XIZ + AZET * ZETAZ
      TXX = -(-4. * UX + 2. * VZ) * CNM / 3.
      TXY = CNM * (UZ + VX)
      TYY = -CNM / 3. * (-4. * VZ + 2. * UX)
      R4  = ((U(I,K)+U(I,K-1))*TXX+(V(I,K)+V(I,K-1))*TXY)*0.5
     1     + CNM / PR/(GAMMA - 1.) * AX
      S4  = ((U(I,K)+U(I,K-1))*TXY+(V(I,K)+V(I,K-1))*TYY)*0.5
     1     + CNM / PR / (GAMMA - 1.) * AZ
      R4 = 0.
      S4 = 0.
      RH1(K) = 0.
      RH2(K) = (ZETAX * TXX + ZETAZ * TXY) / YAC
      RH3(K) = (ZETAX * TXY + ZETAZ * TYY) / YAC
   60 RH4(K) = (ZETAX * R4 + ZETAZ * S4) / YAC
      DO 70 K = 2 , KMAXM1
      DQ1(I,K) = DQ1(I,K) + RH1(K+1) - RH1(K)
      DQ2(I,K) = DQ2(I,K) + RH2(K+1) - RH2(K)
```

```
      DQ3(I,K) = DQ3(I,K) + RH3(K+1) - RH3(K)
      DQ4(I,K) = DQ4(I,K) + RH4(K+1) - RH4(K)
   70 CONTINUE
C
      RETURN
      END
      SUBROUTINE LOAD(CPS,CL,CD,CM,ALFAS)
      COMMON/GRID1/X(161,41),Y(161,41)
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      DIMENSION CPS(161)
C
C     THIS SUBROUTINE COMPUTES THE INVISCID CONTRIBUTIONS
C     TO LOADS ON THE AIRFOIL SURFACE
C
      IMAXM1 = IMAX - 1
      CL = 0.
      CD = 0.
      CM = 0.
      DO 400 I = ITEL , ITEU - 1
      XL = .5 * (X(I,1)+X(I+1,1))
      YL = .5 * (Y(I,1)+Y(I+1,1))
      DX = X(I+1,1) - X(I,1)
      DY = Y(I+1,1) - Y(I,1)
      CPA = CPS(I+1) * .5 + CPS(I) * .5
      DCL = CPA * (-DX)
      DCD = CPA * DY
      CL = CL + DCL
      CD = CD + DCD
  400 CM = CM + DCD * YL - DCL * XL
C
      DCL = CL * COS(ALFAS) - CD * SIN(ALFAS)
      CD = CL * SIN(ALFAS) + CD * COS(ALFAS)
      CL = DCL
      RETURN
      END
      SUBROUTINE WRAP(II,JJ,XSING,YSING,XP,YP,S0,A0,Y0)
      DIMENSION S0(161,4),Y0(41,4),A0(161,4),XP(1),YP(1)
C
C     THIS SUBROUTINE UNWRAPS THE AIRFOIL AND STORES THE UNWRAPPED
C     SURFACE IN ARRAYS A0 AND S0. IT ALSO DETERMINES THE STRETCHING
C     IN THE ETA DIRECTION.
C
      IMID = (II + 1) / 2
      DY   = .8 / (JJ - 2)
      DO 1 J = 2 , JJ
      Y = FLOAT(J-2) * DY
    1 Y0(J,1) = 1.25 * Y / (1. - Y * Y)
      Y0(1 , 1) = - Y0(3,1)
      PI = 4. * ATAN ( 1.)
      ANGL = PI + PI
      U = XP(1) - XSING
      V = YP(1) - YSING
      U = 1.
      V = 0.
      IIM1 = II - 1
      DO 2 I = 1 , II
      X11 = XP(I) - XSING
      Y11 = YP(I) - YSING
      ANGL = ANGL + ATAN2((U*Y11-V*X11),(U*X11+V*Y11))
      R    = SQRT(X11**2 + Y11**2)
```

```
      U   = X11
      V   =Y11
      R = SQRT(R)
      A0(I,1) = R * COS(.5 * ANGL)
    2 S0(I,1) = R * SIN(.5 * ANGL)
C!!!!!IF OUTPUT OF UNWRAPPED COORDINATES IS DESIRED
C      WRITE (6,1000)
C      WRITE (6,2000) (I,A0(I,1),S0(I,1),I = 1 , II)
      RETURN
 1000 FORMAT(1X,'UNWRAPPED COORDINATES IN THE TRANSFORMED PLANE')
 2000 FORMAT(I5 , 2F20.8)
      END
      SUBROUTINE TABINT(XP,YP,XSING,YSING,N)
      DIMENSION XP(161),YP(161),S0(161),A0(161)
C!!!!!SMOOTH THE AIRFOIL SURFACE BY FINDING ADDITIONAL POINTS
      U = XP(1) - XSING
      V = YP(1) - YSING
      U = 1.
      V = 0.
      ANGL = 8. * ATAN(1.)
      DO 1 I = 1,N
      X11 = XP(I) - XSING
      Y11 = YP(I) - YSING
      ANGL = ANGL + ATAN2((U*Y11-V*X11),(U*X11+V*Y11))
      R = SQRT(X11**2 + Y11 ** 2)
      U = X11
      V = Y11
      R = SQRT(R)
      A0(I) = R * COS(ANGL * .5)
    1 S0(I) = R * SIN(ANGL * .5)
      DX =(A0(N)-A0(1))/96.
      A0ST = A0(1)
      DO 3 I = 1 , 97
      XX = FLOAT(I-1) * DX + A0ST
      CALL TAINT(A0,S0,XX,YY,N,3,NER,MON)
      XP(I) = XX * XX - YY * YY + XSING
   3  YP(I) = 2. * XX * YY + YSING
      RETURN
      END
      SUBROUTINE TAINT(XTAB,FTAB,X,FX,N,K,NER,MON)
      DIMENSION XTAB(1),FTAB(1),T(10),C(10)
C
C     NASA - AMES SUBROUTINE FOR POLYNOMIAL INTERPOLATION
C     OF A TABULATED FUNCTION
C
      IF(N-K) 1 , 1 , 2
    1 NER = 2
      RETURN
    2 IF(K-9) 3,3,1
    3 IF(MON) 4,4,5
    5 IF(MON-2) 6,7,4
    4 J = 0
      NM1 = N - 1
      DO 8 I = 1 , NM1
      IF(XTAB(I) - XTAB(I+1)) 9,11,10
   11 NER = 3
      RETURN
    9 J = J-1
      GO TO 8
   10 J = J+1
```

```
    8 CONTINUE
      MON = 1
      IF(J) 12 , 6 , 6
   12 MON = 2
    7 DO 13 I = 1 , N
      IF(X - XTAB(I)) 14,14,13
   14 J = I
      GO TO 18
   13 CONTINUE
      GO TO 15
    6 DO 16 I = 1 , N
      IF(X-XTAB(I)) 16,17,17
   17 J = I
      GO TO 18
   16 CONTINUE
   15 J = N
   18 J = J - (K+1) / 2
      IF(J) 19,19,20
   19 J = 1
   20 M = J + K
      IF(M - N) 21,21,22
   22 J = J - 1
      GO TO 20
   21 KP1 = K + 1
      JSAVE = J
   26 DO 23 L = 1, KP1
      C(L) = X - XTAB(J)
      T(L) = FTAB(J)
   23 J = J+1
      DO 24 J = 1,K
      I = J+1
   25 T(I) = (C(J)*T(I)-C(I)*T(J))/(C(J)-C(I))
      I = I+1
      IF(I-KP1) 25,25,24
   24 CONTINUE
      FX = T(KP1)
      NER = 1
      RETURN
      END
      SUBROUTINE SING(N2,N,X,Z,XLE,YLE,TEA,TES,XSING,YSING,CHD)
C
C
C     THIS SUBROUTINE COMPUTES SINGULAR POINT LOCATIONS.
C
      DIMENSION X(2) , Z(2)
      NU = N2
      N1 = N2 + 1
      N3 = N2 - 1
      X1 = X(N1)
      Z1 = Z(N1)
      X2 = X(N2)
      Z2 = Z(N2)
      X3 = X(N3)
      Z3 = Z(N3)
      D1 = X2 ** 2 - X1 ** 2
      D2 = Z2 ** 2 - Z1 ** 2
      D3 = X2 - X1
      D4 = Z2 - Z1
      D5 = X3 ** 2 - X1 ** 2
      D6 = Z3 ** 2 - Z1 ** 2
```

```
      D7  = X3 - X1
      D8  = Z3 - Z1
      B   = (D7 * ( D1 + D2) - D3*(D5+D6))/(2.*(D7*D4-D3*D8))
      IF(ABS(D3).LT.ABS(D7)) GO TO 10
      A = (D1 + D2 - 2. * B * D4) / (2. * D3)
      GO TO 20
   10 A = (D5 + D6 - 2. * B * D8) / ( 2. * D7)
   20 CONTINUE
      R = SQRT((X2-A)** 2 + (Z2-B)**2)
      XLE = X(NU)
      YLE = Z(NU)
      CHD = X(1) - X(NU)
      A2 = (Z(2)-Z(1)) /(X(2) - X(1))
      A1 = (Z(N)-Z(N-1))/(X(N)-X(N-1))
      TES = .5 * (A1 + A2)
      TEA = A2 - A1
      TEA = TEA * 57.29578
      XSING = (A+XLE) /2.
      YSING = (B+YLE) / 2.
      RETURN
      END
      SUBROUTINE AIRFOL(II,JJ,IT,IE)
      COMMON/GRID1/X(161,41),Z(161,41)
      COMMON/YSYM/ISYM
      DIMENSION S0(161,4),A0(161,4),Y0(41,4),XP(161),YP(161),
     1E(161),F(161),B0(49)
C
      DATA (B0(I),I=1,32)/1.,1.0414,1.0836,1.1270,1.1715,1.2175,1.2651,
     11.3145,1.3659,1.4199,1.4755,1.5349,1.5973,1.6636,1.7342,1.8099,
     21.8914,1.9799,2.0764,2.1829,2.3012,2.4341,2.5653,2.7597,2.9646,
     33.2106,3.5141,3.9019,4.4219,5.1687,6.3632,8.6809/
C
C!!!!!COMPUTE THE COMPUTATIONAL GRID POINTS BASED ON INPUT AIRFOIL SHAPE
      DO 8 I = 1 , 32
    8 A0(I,1) = B0(I)
      READ (5,1)
    1 FORMAT(1X)
      READ (5,2) FNU,FNL,ZSYM
    2 FORMAT(3F10.0)
      ISYM = 0
      IF(ZSYM.NE.0.) ISYM = 1
      II = 157
      JJ =  41
      IT =  31
      IE = 127
      IIP1 = II + 1
      IIM1 = II - 1
      IIJJ = II * JJ
      IIJJ2 = II * ( JJ-2)
      ILE  = (IT + IE ) / 2
      ISTP = 0
      NN   = 5
      NRF  = 0
      NOTAPE = 1
      PI   = 4. * ATAN(1.)
      NU = FNU
      NL = FNL
      N  = NU + NL - 1
      READ(5,1)
      READ (5,333) (XP(I),YP(I),I = NL , N)
```

```
  333 FORMAT(2F10.0)
 9994 FORMAT(F20.8)
      L = N + 1
      IF(ZSYM .GT. 0.) GO TO 9995
      L = NL + 1
      READ(5,1)
      READ (5,333) (XP(L-I),YP(L-I),I=1,NL)
      GO TO 9996
 9995 K1 = L
      DO 16 I = NL , N
      K = K1 - I
      XP(K) = XP(I)
      YP(K) = - YP(I)
   16 CONTINUE
 9996 SCALE = 1. /(XP(1)-XP(NL))
      XX    = XP(NL)
      YY    = YP(NL)
      DO 9997 I = 1 , N
      XP(I) = XP(I) * SCALE
 9997 YP(I) = YP(I) * SCALE
      CALL SING(NU,N,XP,YP,XLE,ZLE,TEA,TES,XSING,YSING,CHD)
      CALL TABINT(XP,YP,XSING,YSING,N)
      NBODY = IE + 1 - IT
      DO 6791 I = 1 , NBODY
      L = I - 1
      E(IT+L) = XP(I)
 6791 F(IT+L) = YP(I)
      IEP1 = IE + 1
      SLOPT = TES * .75
      DO 438 I = IEP1 , II
      I1 = I +1 - IE
      E(I) = A0(I1,1)
      DXI = 1. / 48.
      D   = 4. / 3. * (E(I) - .25)
      F(I) = F(IE) + SLOPT * ALOG(D) / D
      L = IIP1 - I
      E(L) = E(I)
  438 F(L) = F(IT) + SLOPT * ALOG(D)/D
C      WRITE (6,439)
  439 FORMAT(2X,3H  I,19X,1HX,19X,1HY)
C      WRITE (6,37) (I,E(I),F(I),I = 1 , II)
      CALL WRAP(II,JJ,XSING,YSING,E,F,S0,A0,Y0)
C!!!!!MAP GRID BACK TO PHYSICAL PLANE AND SHIFT TO QUARTER CHORD
      DO 10 J = 2 , JJ
      DO 10 I = 1 , II
      X(I,J-1) = A0(I,1)**2 - (S0(I,1)+Y0(J,1))**2
     1 - 0.25
   10 Z(I,J-1) = 2. * A0(I,1) * (S0(I,1)+Y0(J,1))
      JJ = JJ - 1
      RETURN
   37 FORMAT(I5,2F20.8)
      END
      SUBROUTINE CLUSTR(DS)
      COMMON/GRID1/X(161,41),Z(161,41)
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      DIMENSION S(41),XP(41),YP(41),R(41)
C
C      THIS SUBROUTINE CLUSTERS A GIVEN X,Z GRID SUCH THAT THE FIRST POINT IS AT
C      THE USER-SPECIFIED DISTANCE DNMIN
C!!!!!COMPUTE THE OLD STRETCHING
```

```
      DO 100 I = 1 , IMAX
      S(1) = 0.
      XP(1) = X(I,1)
      YP(1) = Z(I,1)
      DO 10 K = 2 , KMAX
      XP(K) = X(I,K)
      YP(K) = Z(I,K)
   10 S(K) = SQRT((XP(K)-XP(K-1))**2+(YP(K)-YP(K-1))**2)
     1+S(K-1)
      SUMDX = S(KMAX)
      CALL STRTCH(SUMDX,DS,F1,KMAX,FACTOR)
C      WRITE (6,200) I,FACTOR
      R(1) = 0.
      DR = DS
      DO 20 K = 2 , KMAX
      R(K) = R(K-1) + DR
      DR = DR * FACTOR
   20 CONTINUE
      RLAST = 1. / R(KMAX)
      DO 30 K = 2 , KMAX
      R1 = R(K) * RLAST * S(KMAX)
C!!!!!REDISTRIBUTE THE CONSTANT-ETA LINES
      CALL TAINT(S,XP,R1,XP1,KMAX,3,NER,MON)
      X(I,K) = XP1
      CALL TAINT(S,YP,R1,YP1,KMAX,3,NER,MON)
      Z(I,K) = YP1
   30 CONTINUE
  100 CONTINUE
C      WRITE (6,115)
      DO 110 I = 1 , IMAX
      DX = X(I,2) - X(I,1)
      DY = Z(I,2) - Z(I,1)
      DN = SQRT(DX*DX+DY*DY)
C      WRITE(6,120) I , DX , DY , DN
  110 CONTINUE
      RETURN
  115 FORMAT(5X,6HNORMAL,1X,8HDISTANCE,3H AT,4H THE,5H WALL,/
     1,5H     I,8X,2HDX,8X,2HDY,8X,2HDN,//)
  120 FORMAT(I5,3F10.5)
  200 FORMAT(I5,F10.5)
      END
      SUBROUTINE STRTCH(SUMDX,DX1,F1,N1,R)
C
C     THIS SUBROUTINE DETERMINES A GEOMETRIC
C     PROGRESSION OF GRID SPACING BETWEEN 1 AND N1 SUM THAT
C     SUM(DX) EQUALS SUMDX. THE RATIO BETWEEN SUCCESSIVE
C     SPACINGS IS R.
      N = N1 - 1
      R = 1.5
      E1 = 1.E-04
      E2 = 1.E-04
      DO 10 L = 1, 50
      F= (R-1) * SUMDX - DX1*(R**N-1)
      FP = SUMDX - DX1 * FLOAT(N) * R ** (N-1)
      RITER = R - F/ FP
C     IF(1.E-02.LT.RITER.AND.RITER.LT.1.) RITER = 1.
C     IF(1..LT.RITER.AND.RITER.LT.100.) RITER=.01
      IF(ABS(F-RITER).LT. R*E1) GO TO 1
      R = RITER
   10 CONTINUE
```

```
      R = 1.0001
      DX1 = DZTOT/FLOAT(N1-1)
      RETURN
    1 R= RITER
      RETURN
      END
      SUBROUTINE EDDY(DALFA)
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      COMMON/MUTUR/CMU(161,41)
      COMMON/SKINCF/CF(161)
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      COMMON/GRID1/X(161,41),Z(161,41)
      DIMENSION TIN(41),TOUT(41),Y(41)
C
C       INITIALIZE VISCOSITY EVERYWHERE
      FACT1 = DT * AMINF / REYREF
      CMUMAX = 100. * FACT1 / DT
      DO 1 K = 1 , KMAX
      DO 1 I = 1 , IMAX
    1 CMU(I,K) = FACT1
C     THIS SUBROUTINE COMPUTES THE EDDY VISCOSITY BASED ON THE
C     BALDWIN-LOMAX TWO LAYER MODEL
C
      DO 100 I = 2 , IMAX - 1
      UDIF = 0.
      FMAX = 0.1E-06
      YMAX = .1E-06
      FYMAX = 0.
      Y(1) = 0.
      UWALL = 0.
      IF(I.GT.ITEU.OR.I.LE.ITEL)UWALL = SQRT(Q2(I,1)**2+Q3(I,1)**2)/
     1Q1(I,1)
C     COMPUTE TAU AT THE WALL
      UET = 1.*(Q2(I,2)/Q1(I,2) - Q2(I,1)/Q1(I,1))
      VET = 1.*(Q3(I,2)/Q1(I,2) - Q3(I,1)/Q1(I,1))
      XXI = X(I+1,1) - X(I-1,1)
      ZXI = Z(I+1,1) - Z(I-1,1)
      XET = 4. * X(I,2) - 3. * X(I,1) - X(I,3)
      ZET = 4. * Z(I,2) - 3. * Z(I,1) - Z(I,3)
      XXI = .5 * XXI
      ZXI = .5 * ZXI
      XET = .5 * XET
      ZET = .5 * ZET
      YAC = 1. / (XXI * ZET - ZXI * XET)
      OMEGA =  (UET * XXI - VET * ZXI ) * YAC
      TWALL = AMINF * OMEGA / REYREF
      CF(I) = 2. * TWALL / (AMINF**2)
      FACT = SQRT(Q1(I,1) * ABS(TWALL))*REYREF/(26.*AMINF)
      DO 10 K = 2 , KMAX-1
      UXI = (Q2(I+1,K)/Q1(I+1,K) - Q2(I-1,K)/Q1(I-1,K))
      VXI = (Q3(I+1,K)/Q1(I+1,K)-Q3(I-1,K)/Q1(I-1,K))
      UET = (Q2(I,K+1)/Q1(I,K+1)-Q2(I,K-1)/Q1(I,K-1))
      VET = (Q3(I,K+1)/Q1(I,K+1)-Q3(I,K-1)/Q1(I,K-1))
      XXI = X(I+1,K) - X(I-1,K)
      ZXI = Z(I+1,K) - Z(I-1,K)
      XET = X(I,K+1) - X(I,K-1)
      ZET = Z(I,K+1) - Z(I,K-1)
      YAC = 1. / (XXI * ZET - ZXI * XET)
      OMEGA = ABS(UET*XXI+VET*ZXI-UXI*XET-VXI*ZET) * YAC
```

```
      UDIF = SQRT(Q2(I,K)**2+Q3(I,K)**2)/Q1(I,K) - UWALL
      IF(ABS(UDIF).GT.UDIFMAX) UDIFMAX = ABS(UDIF)
      Y(K) = SQRT((X(I,K)-X(I,K-1))**2+(Z(I,K)-Z(I,K-1))**2)+Y(K-1)
      F = Y(K) * OMEGA
      IF((Y(K)*FACT).GT.20.) GO TO 31
      IF(I.GT.ITEL.AND.I.LT.ITEU) F = F * (1. - EXP(-Y(K)*FACT))
   31 CONTINUE
C
C     MODIFIED TURBULENCE MODEL APPLY FOR SPECIFIED RANGE OF ANGLES WHERE
C     FY IS USED TO FIND THE SECOND PEAK VALUE OF F FUNCTION
C
      IF(ALFA.LT.ALFAI.AND.DALFA.GE.0.) THEN
        FY = F * Y(K)
        IF(FY.GT.FYMAX) THEN
          FYMAX = FY
          FMAX  = F
          YMAX  = Y(K)
        ENDIF
      ENDIF
      IF(ALFA.GE.ALFAI.OR.DALFA.LT.0.) THEN
        IF(F.GT.FMAX) THEN
          FMAX = F
          YMAX = Y(K)
        ENDIF
      ENDIF
      FCT = Y(K) * FACT
      IF(FCT.GT.20.) FCT = 20.
      FCT = ABS(FCT)
      EL = .4 * Y(K) * (1. - EXP(-FCT))
      TIN(K) = Q1(I,K) * EL * EL * OMEGA
      TIN(K) = ABS(TIN(K))
   10 CONTINUE
      KSWTCH = 0
      FWAKE = YMAX * FMAX
      F1 = 0.25 * YMAX * UDIF **2 / FMAX
      IF(F1.LT.FWAKE) FWAKE = F1
      DO 20 K = 2 , KMAX - 1
      FKLEB = 0.
      IF(ABS(Y(K)/YMAX).LT.1.E+04) THEN
      FKLEB = 1. / (1. + 5.5 * (0.3 * Y(K)/YMAX) ** 6)
      END IF
      TOUT(K) = .0168 * 1.6 * Q1(I,K) * FWAKE * FKLEB
      TOUT(K) = ABS(TOUT(K))
      IF(KSWTCH.NE.0) GO TO 20
      IF(TIN(K).GT.TOUT(K)) KSWTCH = K - 1
   20  CONTINUE
C!!!!!TOTAL VISCOSITY IS SUM OF LAMINAR AND INNER/OUTER LAYER AS APPROPRIATE
      DO 30 K = 2 , KMAX - 1
      IF(K.LE.KSWTCH) THEN
      CMU(I,K) = DT * (AMINF/REYREF + ABS(TIN(K)))
      ELSE
      CMU(I,K) = DT * (AMINF / REYREF + ABS(TOUT(K)))
      END IF
   30 CONTINUE
  100 CONTINUE
      RETURN
      END
      SUBROUTINE RESI(OMEGA)
      COMMON/PERTR/DQ1(161,41),DQ2(161,41),DQ3(161,41),DQ4(161,41)
      COMMON/GRID1/X(161,41),Z(161,41)
```

```
      COMMON/DGRID/DT,IMAX,KMAX,ITEL,ITEU
      COMMON/FLOW/Q1(161,41),Q2(161,41),Q3(161,41),Q4(161,41)
      COMMON/PAR/GAMMA,REYREF,ALFA,ALFA1,REDFRE,AMINF,ALFAI
      DIMENSION RHS(161,4)
      XTAU(I,K) = OMEGA * Z(I,K)
      YTAU(I,K) = - OMEGA * X(I,K)
C     THIS SUBROUTINE COMPUTES THE RESIDUAL ON THE RIGHT HAND
C     SIDE ARISING FROM THE EULER- PART OF THE GOVERNING EQUATIONS
C
C    FLUX TERMS WITHIN THE XI- DERIVATIVE
      DO 100 K = 2 , KMAX - 1
      DO 10 I = 1 , IMAX
      UCON = (Q2(I,K)/Q1(I,K) ) * (Z(I,K+1)-Z(I,K-1))
     1 - (Q3(I,K)/Q1(I,K)) * (X(I,K+1)-X(I,K-1))
      UCON = 0.25 * DT * UCON
      XIT = - XTAU(I,K) *(Z(I,K+1)-Z(I,K-1))
     1 + YTAU(I,K) * (X(I,K+1) - X(I,K-1))
      XIT = XIT * DT * 0.25
      UCON = UCON + XIT
      RHS(I,1) = UCON * Q1(I,K)
      R = 1. / Q1(I,K)
      P = (GAMMA-1.) * (Q4(I,K) - .5 * R *(Q2(I,K)**2+
     1                                     Q3(I,K)**2))
      RHS(I,2) = Q2(I,K) * UCON + P * DT * 0.25 * (Z(I,K+1) - Z(I,K-1))
      RHS(I,3) = Q3(I,K) * UCON - P * DT * 0.25 * (X(I,K+1)-X(I,K-1))
      RHS(I,4) = UCON * (Q4(I,K)+P) - XIT * P
   10 CONTINUE
      DO 11 I = 2 , IMAX - 1
      DQ1(I,K) = DQ1(I,K) - RHS(I+1,1) + RHS(I-1,1)
      DQ2(I,K) = DQ2(I,K) - RHS(I+1,2) + RHS(I-1,2)
      DQ3(I,K) = DQ3(I,K) - RHS(I+1,3) + RHS(I-1,3)
   11 DQ4(I,K) = DQ4(I,K) - RHS(I+1,4) + RHS(I-1,4)
  100 CONTINUE
C
C      FLUX TERMS WITHIN THE ETA- DERIVATIVE
C
       DO 200 I = 2 , IMAX - 1
      DO 20 K = 1 , KMAX
      VCON = (Q2(I,K)/Q1(I,K) ) * (Z(I-1,K)-Z(I+1,K))
     1      +(Q3(I,K)/Q1(I,K) ) * (X(I+1,K)-X(I-1,K))
      VCON = VCON * 0.25 * DT
      ETAT = -XTAU(I,K) * (Z(I-1,K)-Z(I+1,K)) - YTAU(I,K)*
     1                                          (X(I+1,K)-X(I-1,K))
      ETAT = ETAT * 0.25 * DT
      VCON = VCON + ETAT
      RHS(K,1) = VCON * Q1(I,K)
      P = (GAMMA-1.) * (Q4(I,K) - 0.5 *(Q2(I,K)**2+Q3(I,K)**2)/Q1(I,K))
      RHS(K,2) = VCON * Q2(I,K) +P * DT * .25 * (Z(I-1,K)-Z(I+1,K))
      RHS(K,3) = VCON * Q3(I,K) + P * DT * .25 * (X(I+1,K) - X(I-1,K))
      RHS(K,4) = VCON * (Q4(I,K)+P) - ETAT * P
   20 CONTINUE
      DO 21 K = 2 , KMAX - 1
      DQ1(I,K) = DQ1(I,K) - RHS(K+1,1) + RHS(K-1,1)
      DQ2(I,K) = DQ2(I,K) - RHS(K+1,2) + RHS(K-1,2)
      DQ3(I,K) = DQ3(I,K) - RHS(K+1,3) + RHS(K-1,3)
   21 DQ4(I,K) = DQ4(I,K) - RHS(K+1,4) + RHS(K-1,4)
  200 CONTINUE
      RETURN
      END
      SUBROUTINE ROTGRID(X,Z,IMAX,KMAX,DALFA)
```

```
C     ROTATE GRID IN THE CLOCKWISE DIRECTION BY AN AMOUNT DALFA
      DIMENSION X(161,41),Z(161,41)
      CA = COS(DALFA)
      SA = - SIN(DALFA)
      DO 20 K = 1 , KMAX
      DO 20 I = 1 , IMAX
      X1 = X(I,K)
      Z1 = Z(I,K)
      X(I,K) = X1 * CA - Z1 * SA
   20 Z(I,K) = Z1 * CA + X1 * SA
      RETURN
      END
      SUBROUTINE CPPLOT(I1,I2,FMACH,X,Y,CP)
C
C     THIS SUBROUTINE PLOTS CP AT EQUAL INTERVALS IN THE MAPPED PLANE
C
      DIMENSION KODE(4),LINE(90),X(161),Y(161),CP(161)
      DATA KODE/1H ,1H+,1HI,1H*/
      WRITE ( 6 , 2)
    2 FORMAT(50H0PLOT OF CP AT EQUAL INTERVALS IN THE MAPPED PLANE/
     1       10H0      X    ,10H      X/C  ,10H      CPL  ,10H      CPU  )
      CP0 = (1. + .2 * FMACH **2) ** 3.5 - 1.
      CP0 = CP0 / ( .7 * FMACH **2)
      K0 = 30. * CP0 + 4.5
      IMIN = (I2-I1)/2 + I1
      ILOW = 2 * IMIN
      CHD=X(I1) - X(IMIN)
      DO 12 I = 1 , 90
   12 LINE(I) = KODE(1)
      DO 34 I = IMIN  , I2
      K = 30. * (CP0 - CP(I)) + 4.5
      K1 = 30. * (CP0 -CP(ILOW-I)) + 4.5
      IF(K.LT.1) K = 1
      IF(K1.LT.1) K1 = 1
      IF(K.GT.90) K = 90
      IF(K1 .GT. 90) K1 = 90
      LINE(K0) = KODE(3)
      LINE(K) = KODE(2)
      LINE(K1) = KODE(4)
      XOC = (X(I) - X(IMIN)) / CHD
      WRITE (6,610) X(I),XOC,CP(ILOW-I),CP(I),LINE
      LINE(K1) = KODE(1)
   34 LINE(K) = KODE(1)
      RETURN
  610 FORMAT(4F10.4,90A1)
      END
/EOF
 NACA 0012 AIRFOIL,  RUN:
  159   41      .0025          5.     15.000      10.00      0.00   0.151     .5000
    NO. OF STEPS
4500.
   REYNOLDS NUMBER IN MILLIONS, DISTANCE OF FIRST POINT OFF THE WALL
  .345      .00005
 TSTART
  0.0
 FULOUT
 -1.0
RESTART,PITCH,PLUNGE,OUTPUT OPTIONS SPECIFIED IN THE NEXT CARD
TRUE TRUE FALSE
  FNU       FNL          FSYM
```

33.	33.	1.
X	Y	
0.	0.	
0.0050	.01153	
.0125	.01894	
.0250	.02615	
.0500	.03555	
.0750	.04200	
.1000	.04683	
.1500	.05345	
.2000	.05737	
.2500	.05941	
.2600	.05962	
.2700	.05978	
.2800	.05992	
.2900	.05999	
.3000	.06000	
.3100	.05999	
.3200	.05992	
.3300	.05980	
.3400	.05965	
.3500	.05947	
.4000	.05803	
.4500	.05581	
.5000	.05294	
.5500	.04952	
.6000	.04563	
.6500	.04137	
.7000	.03664	
.7500	.03161	
.8000	.02623	
.8500	.02055	
.9000	.01448	
.9500	.00807	
1.0000	.00126	

APPENDIX B
NOTES ON USE OF THE NAVIER-STOKES SOLVER

1. JOB CARDS

The JCL options are selected by removing the "comment" designator ("*.") at the beginning of the applicable lines. The options available are:

- Save the current solution. Values of TIME, Q1, Q2, Q3, and Q4 are saved (logical unit 08) for subsequent restart. Activate the two lines referencing "NEWSLN".
- Create pressure coefficient data file. Data is accumulated through the runs to be accessed and read by a separate program. Activate the "OLDCP" statements to access and add to previously stored data. Activate "NEWCP" statements to store current cumulative data. A new version is created each time, so the files must be purged periodically.
- Start from a stored solution. The above values are read (logical unit 07) and iteration continued from that point. Activate the two lines referencing "OLDSLN".
- Creat PLOT3D files. Conversion from Cray to VAX binary is handled and properly formatted "Q" and "XYZ" files are created for use with the PLOT3D graphics program. Activate the Q and XYZ "ASSIGN" statements and the "SENDVAX" and "ACCESS" statements ***before*** the "EXIT" card.
- Create "troubleshooting" PLOT3D files. If the solution "blows up" and the appropriate "WRITE" statements are not commented out in the main program, Q and XYZ files are created to investigate the status of the solution at the last "successful" iteration. Activate the "ASSIGN" statements and the "ACCESS" and "SENDVAX" statements ***after*** the "EXIT" card.
- Use job chaining. The FETCH, REWIND, and SUBMIT statements are used to call up another program when current run is completed.

In addition, if it is desired to save more than one solution, the PDN (5 digits) and ID may be changed. Old data sets must be purged from Cray. This is accomplished by placing an "AUDIT." card after the account card when running a program or by running "AUDIT.JCL" to obtain a listing of current data sets. Then, on VAX, run SKILLJCL to create a JCL to delete Cray PDN's. It is then only

necessary to precede this JCL by a JOB card and ACCOUNT card and run the program as usual. Consult the ACF Cray Users' Guide for detailed information on job control cards.

2. MAIN PROGRAM

Certain changes may be made within the main program. These changes affect the program execution or change the output.

- The frequency of steady-state output may be changed by varying the value in the first statement after the comment "FOR STEADY STATE OUTPUT USE THE FOLLOWING".
- The interval for exiting the program (in order to save a solution and generate PLOT3D files) may be changed under "MULTIPLY NCPOUT...". For example: NEXIT = 24 * NCPOUT means the program will exit automatically four times a cycle, or every 24 times the normal printed output is generated.
- Velocity profile information may be output if desired (as when a permanent record of a converged solution is desired) by activating the WRITE statement just before the CONTINUE statement numbered 4000. This outputs the indices of the X and Z coordinates of each grid point with the corresponding values of ρu, ρv, eddy viscosity, total velocity ($\sqrt{u^2 + v^2}$), and distance normal to the wall.
- At the beginning of subroutine SLPS, the value of the implicit damping coefficient may be adjusted by changing the multiple of WW.
- At the end of subroutine SLPS, the frequency with which the residuals are output may be set by changing the value under "SELECT THE INTERVAL...". At least every ten steps is recommended.
- Other output values may be turned on and off as well. "UNWRAPPED COORDINATES IN THE TRANSFORMED PLANE" is in subroutine "WRAP". Airfoil coordinates are in "AIRFOL". Grid spacing and "NORMAL DISTANCE AT THE WALL" are in "CLUSTR".

3. DATA CARDS

Most of the "tuning" of the program is done with the data cards:

- 1st Line - Name of airfoil and other identifying information. Eighty characters available.

- 2nd Line - (1) and (2) The first two values, IMAX and KMAX (format I5), are the number of X and Z coordinate locations to be used. These remain at 159 and 41 for the 161×41 C-grid used at present. The remainder of the line contains seven values in format F10.0. (3) DT: size of the time step. This is automatically set to 1.0 within the program when the reduced frequency is less than or equal to 0.001. In this case the program uses the local time-step option (relaxation). For dynamic stall DT = 0.005 is recommended, at angles below 20 degrees, smaller at higher angles. (4) WW: explicit artificial viscosity term. Normally 2-5, with about 2-3 recommended for static cases, 5 for dynamic stall. Higher numbers have greater effect on solution. (5) ALFA: mean angle of attack. (6) ALFA1: amplitude of oscillation. (7) ALFAI: angle below which a modified turbulence model is used (upstroke only) to compute eddy viscosity (Baldwin-Lomax model). Normally set to 19 degrees for dynamic stall. May affect stability of solution. (8) REDFRE: reduced frequency. (9) AMINF: Mach number.
- 3rd Line - "NO. OF STEPS"--not read.
- 4th Line - FNSTP: number of time steps to be done on the present run (format F10.4).
- 5th Line - "REYNOLDS NUMBER..."--not read.
- 6th Line - (1) REYREF: the Reynolds number in millions (format F10.4). A negative value means inviscid flow. (2) DNMIN: distance of the first point off the wall (format F10.4). For Reynolds numbers up to 3 million 0.00005 should be used normally. Stability of the solution may be improved by increasing this value in some cases (ie., high AOA steady angle of attack).
- 7th Line - "TSTART"--not read.
- 8th Line - TSTART (format F10.4): time calculations have been advanced up to the previous run. When a negative value is used, TSTART is read from logical unit 08 (see JCL comments). Then normally use 0.0 for initial runs and −1.0 for restarts. Must use 0.0 for first dynamic run from converged steady-state solution.
- 9th Line - "FULOUT"--not read.
- 10th Line - FULOUT (format F10.4): −1.0 means no plotting files will be generated. Set 0.0 to begin full output, then set 1.0 to continue. When using full output, the appropriate job cards must be activated.
- 11th Line - "RESTART, PITCH,..."--not read.

- 12th Line - RSTRT, PITCH, PLUNGE (format L5): If RSTRT is set, stored values of TIME, Q1, Q2, Q3, and Q4 are read to continue iteration. PITCH is set for dynamic stall and indicates change in angle of attack. PLUNGE will not be set for present purposes. It indicates up and down motion of the airfoil.
- 13th Line - The remaining lines define airfoil geometry and for the present are set for the NACA0012 airfoil.

4. ADDITIONAL NOTES

For high angle of attack (separated flow), or if otherwise desired for time accurate steady-state calculations, use reduced frequency of 0.002 and ALFAI (amplitude) = 0. Set PITCH = FALSE and DT = 0.005, as for dynamic stall. For stability WW should be set to 5 (or even higher, up to 10) and ALFAI at or below the minimum angle to use the original Baldwin-Lomax turbulence model. The distance of the first point off the wall may also be increased ($\sim$ 0.0001), which has the effect of coarsening the grid for the troublesome fine-mesh leading-edge area.

When doing dynamic stall simulations where the AOA goes to the 20-25 degrees range, use the DT = 0.005 at lower angles for computation time, but reduce this value when restarting going into the high angle portion of the cycle.

The values of DRMAX, DUMAX, etc. may be useful when a solution "blows up". The IR and KR values give the indices of the X - Z location on the grid where density changed the fastest (IR = 80 is the leading edge). Normally, this will be near the leading edge. DT should then be reduced.

Reynolds numbers of 10^6 to 10×10^6 should show no effect on sensitivity of solution for distance of the first point off the wall of .00005, since this places several points in the boundary layer.

Mach numbers of .1-.5 should not alter calculation time significantly, but perhaps twice as many iterations may be required for convergence in the .5-.8 range.

LIST OF REFERENCES

1. American Institute of Aeronautics and Astronautics Paper 85-1769-CP, *Dynamic Stall Progress in Analysis and Prediction*, by L. W. Carr, 19-21 August 1985.

2. American Institute of Aeronautics and Astronautics Paper 81-1289, *Dynamic Stall of NACA 0012 Airfoil in Turbulent Flow - Numerical Study*, by Y. Tassa and N. L. Sankar, June 1981.

3. Gangwani, Santu T., "Synthesized Airfoil Data Method for Prediction of Dynamic Stall and Unsteady Airloads," *Vertica*, v. 8, no. 2, pp. 93-118, 1984.

4. Shamroth, Stephen J., *A Navier-Stokes Calculation of the Airfoil Dynamic Stall Process*, paper presented at AFOSR/FJSRL/ University of Colorado Workshop on Unsteady Separated Flows, 10-11 August 1983.

5. American Institute of Aeronautics and Astronautics Paper 84-0049, *Application of Thin-Layer Navier-Stokes Equations Near Maximum Lift*, by W. K. Anderson, J. L. Thomas, and C. L. Rumsey, 1984.

6. American Institute of Aeronautics and Astronautics Paper 85-0129, *Numerical Solution of Unsteady Viscous Flow Past Rotor Sections*, by N. L. Sankar and W. Tang, 14-17 January 1985.

7. Tang, Wei, *Numerical Solutions of Unsteady Flow Past Rotor Sections*, Ph.D. Thesis, Georgia Institute of Technology, Atlanta, Georgia, August 1986.

8. National Aeronautics and Space Administration Memorandum 84245, *An Experimental Study of Dynamic Stall on Advanced Airfoil Sections*, by W. J. McCroskey, K. W. McAlister, L. W. Carr, and S. L. Pucci, July 1982.

9. Steger, Joseph L., "Implicit Finite-Difference Simulation of Flow about Arbitrary Two-Dimensional Geometries," *AIAA Journal*, v. 16, no. 7, pp. 679-686, July 1978.

10. Anderson, Dale A., Tannehill, J. C., and Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer*, McGraw-Hill Book Company, 1984.

11. American Institute of Aeronautics and Astronautics and Astronautics Paper 80-0010, *Reynolds Number and Compressibility Effects on Dynamic Stall of a NACA 0012 Airfoil*, by N. L. Sankar and Y. Tassa, January 1980.

12. MacCormack, R. W., *An Introduction and Review of the Basics of Computational Fluid Dynamics*, prepared for the AIAA Invited Lecture Series on Computational Fluid Dynamics, Mountain View, California, March 1984.

13. Moran, Jack, *An Introduction to Theoretical and Computational Aerodynamics*, John Wiley and Sons, Inc., 1984.

14. Chakravarthy, Sukumar R., *A Concise Overview of Numerical Methods for the Solution of the Equations of Fluid Dynamics*, prepared for the AIAA Invited Lecture Series on Computational Fluid Dynamics, Mountain View, California, March 1984.

15. American Institute of Aeronautics and Astronautics Paper 82-0228, *Euler Equations--Implicit Schemes and Implicit Boundary Conditions*, by S. Chakravarthy, June 1982.

16. American Institute of Aeronautics and Astronautics Paper 81-1259, *Numerical Solution of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time Stopping Schemes*, by A. Jameson, W. Schmit, and E. Turkel, June 1981.

17. Shang, J. S., "An Assessment of Numerical Solutions of the Compressible Navier-Stokes Equations," *Journal of Aircraft*, v. 22, no. 5, pp. 353-370, May 1985.

18. National Aeronautics and Space Administration Technical Memorandum 86830, *Separated Transonic Airfoil Flow Calculations with a Nonequilibrium Turbulence Model*, by L. S. King and D. A. Johnson, November 1985.

19. Wu, Jiunn-Chi, Kaza, R. V., and Sankar, L. N., *A Technique for the Prediction of Airfoil Flutter Characteristics in Separated Flows*, abstract submitted to the AIAA Dynamics Specialists Conference, Monterey, California, 9-10 April 1987.

20. American Institute of Aeronautics and Astronautics Paper 87-0418, *A Comparison of Turbulence Closure Models for Transonic Flows about Airfoils*, by Lyndell S. King, 12-15 January 1987.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Satyanarayana Bodapati Fluid Mechanics Laboratory Mail Stop 260-1 NASA Ames Research Center Moffett Field, California 94035	5
4.	Chief Fluid Mechanics Laboratory Mail Stop 260-1 NASA Ames Research Center Moffett Field, California 94035	3
5.	Lawrence W. Carr Fluid Mechanics Laboratory Mail Stop 260-1 NASA Ames Research Center Moffett Field, California 94035	2
6.	Chairman Department of Aeronautical Engineering, Code 67 Naval Postgraduate School Monterey, California 93943	5
7.	L. N. Sankar School of Aerospace Engineering Georgia Institute of Technology Atlanta, Georgia 30332	1
8.	Commanding Officer Naval Plant Representative Office McDonnell Douglas Corporation P. O. Box 516 St. Louis, Missouri 63166-0516	1
9.	Curricular Officer Department of Aeronautical Engineering, Code 31 Naval Postgraduate School Monterey, California 93943	1
10.	R. E. Singleton Director, Engineering Sciences U. S. Army Research Office P. O. Box 12211 Research Triangle Park, North Carolina 27709	1